Escola Técnica Superior de Enxeñaría
Universidade de Santiago de Compostela

FaceSampler

Selección automática de mostras do usuario para un sistema non colaborativo de verificación de rostros

Autor:
Fernando Estévez Casado

Titores:
Xosé Manuel Pardo López
Roberto Iglesias Rodríguez

Grao en Enxeñaría Informática

Xullo 2017

Traballo de Fin de Grao presentado na Escola Técnica Superior de Enxeñaría da Universidade de Santiago de Compostela para a obtención do Grao en Enxeñaría Informática

D. Xosé Manuel Pardo López, Profesor do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela, e **D. Roberto Iglesias Rodríguez**, Profesor do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela,

INFORMAN:

Que a presente memoria, titulada *FaceSampler: Selección automática de mostras do usuario para un sistema non colaborativo de verificación de rostros*, presentada por **D. Fernando Estévez Casado** para superar os créditos correspondentes ao Traballo de Fin de Grao da titulación de Grao en Enxeñaría Informática, realizouse baixo a nosa dirección no Departamento de Electrónica e Computación da Universidade de Santiago de Compostela.

E para que así conste aos efectos oportunos, expiden o presente informe en Santiago de Compostela, a 6 de xullo de 2017:

O titor,	O cotitor,	O alumno,
Xosé M. Pardo López	Roberto Iglesias Rodríguez	Fernando Estévez Casado

A Rubén e Glauco, pola paciencia que tiveron escoitando os meus problemas a diario e por ser os primeiros voluntarios en probar a aplicación.

A Xosé, Roberto e Carlos, por confiar en min e adicarme o seu tempo sempre que o necesitei.

Á miña familia, por apoiarme en todas as decisións que levo tomado no meu camiño profesional e por axudarme tanto nos momentos difíciles.

Índice xeral

Índice de figuras

Índice de táboas

Capítulo 1

Introdución

Comparado con outros métodos biométricos como os de recoñecemento de retina, iris ou pegada dactilar, a identificación de caras pode realizarse sen cooperación do suxeito e sen intrusión. O emprego do recoñecemento facial para o control de acceso en dispositivos móbiles leva gañado unha importancia notoria nos últimos anos. Ademais, están emerxendo novas aplicacións deste tipo de identificación, como os pagos móbiles e outras con implicacións legais e de seguridade. En moitas destas aplicacións, ou non é realista agardar colaboración do usuario, ou nin tan sequera é desexable. Polo tanto, a verificación hai que facela de forma totalmente transparente para o usuario do dispositivo móbil. Isto implica que o sistema non pode impoñer restricións en canto ás condicións de captación das imaxes, entre elas a perspectiva coa que se realicen as tomas, que serán moi variadas debido tanto á diversidade de hábitos dos diferentes usuarios, como ás condicións de uso dun mesmo usuario ao longo do tempo.

O contexto do presente proxecto vén marcado por unha liña de investigación do Centro Singular de Investigación en Tecnoloxías da Información da Universidade de Santiago de Compostela (CiTIUS) sobre a verificación de usuarios de dispositivos móbiles que permita distinguilos de posibles intrusos, e onde se considera como condición esencial de usabilidade que a verificación se leve a cabo de xeito totalmente automático e transparente ao usuario. Para abordar este problema débese traballar nun entorno sen restricións e sen contar coa participación activa do suxeito [1]. O termo "sen restricións" fai referencia á independencia tanto do proceso de captura (iluminación, sensor, enfoque ou compresión), como do individuo (orientación do rostro, expresión ou oclusións). O presente proxecto céntrase nunha parte esencial dese sistema de verificación non colaborativo que consiste en seleccionar o instante coas condicións axeitadas para facer a toma automática das imaxes das caras, tanto para a construción do modelo do usuario como para a súa verificación.

Usualmente, nun sistema de recoñecemento facial común, tómase unha imaxe do rostro do individuo e procésase para obter os seus puntos característicos. Logo, contrástase esta información cun modelo dese suxeito, o que permite clasificar a cara como pertencente á persoa ou non. Para obter dito modelo, sen embargo, precísase dispor dun número suficiente de mostras de adestramento da cara do individuo, que permitan construír unha codificación do rostro o máis robusta e discriminante posible. Esta é a razón pola que se foron creando nos últimos anos grandes bancos de datos de imaxes anotadas de acceso público como LFW [2], FDDB [3] ou YTF [4]. Sen embargo, só se construíron uns poucos bancos de datos de acceso público para o recoñecemento biométrico no ámbito dos dispositivos móbiles, como MOBIO [5] ou o de PASC [6], o que impediu o avance nesta liña. Pero, o que é peor, as imaxes provintes deste tipo de bancos soen estar moi nesgadas, isto é, existen diferenzas significativas entre elas.

A captación de imaxes mediante cámaras en dispositivos móbiles está suxeita a características específicas non só a nivel de sensores (escala, resolución, etc.), senón tamén relacionadas coa forma de usar tales dispositivos (orientación, expresión, iluminación, etc.). Todo isto esixe que, para acadar o mellor rendemento, se deba dispor de mostras de caras para a aprendizaxe específicas do contexto. Necesítase, porén, unha ferramenta que poida ser instalada polo individuo no seu móbil e que o "acompañe" no seu día a día. Esta ferramenta debe traballar sen a colaboración do suxeito, pero debe saber cando capturar imaxes para obter boas mostras do seu rostro. Para elo, compre que se axude da información relativa ao estado do dispositivo: orientación, iluminación ambiental, manipulación por parte do usuario, etc.

A través deste traballo preténdese apoiar esta liña de investigación, en particular no relativo á aprendizaxe automática (no propio dispositivo móbil) para a captación das caras dos usuarios. Preséntase así FaceSampler, unha aplicación Android que implementa a proposta anterior e ten unha dobre utilidade. Por un lado, funciona como ferramenta *stand-alone* que permite obter as mostras dos rostros dos usuarios. Polo outro, nun futuro poderían aproveitarse as súas funcionalidades para traballar como mecanismo de selección de mostras de cada usuario durante todo o tempo de vida dunha aplicación de verificación de caras.

1.1. Obxectivos do proxecto

Preséntanse a continuación os obxectivos do proxecto. Compre diferenciar entre obxectivos xerais e específicos.

Obxectivo xeral

Identifícase un único obxectivo xeral, que se formula en base á motivación e á finalidade principal do proxecto.

Obxectivo OBX-1	
Nome:	Elaboración dun software para a captura de rostros.
Versión:	1.0
Descrición:	Desenvolver un software para a captura de rostros de usuario en dispositivos móbiles nun entorno non colaborativo que permita xerar un conxunto de mostras de adestramento de calidade para a posterior construción dun modelo que se poida empregar na fase posterior de identificación.
Sub-obxectivos:	[OBX-2], [OBX-3]
Importancia:	Vital
Urxencia:	Alta
Estado:	Cumprido
Estabilidade:	Alta

Obxectivos específicos

Derívanse do obxectivo xeral e pretenden concretalo, sinalando o camiño a seguir para conseguilo. Indican os efectos específicos que se queren acadar aínda que non expliciten accións directamente medibles.

Obxectivo OBX-2	
Nome:	Recoñecemento facial.
Versión:	1.0
Descrición:	Lograr que a aplicación móbil sexa capaz de tomar imaxes do usuario do dispositivo sen a participación activa do mesmo, a partir desas imaxes, discriminar entre aquelas que presenten un rostro humano e aquelas nas que non se de esta situación.
Sub-obxectivos:	Non aplica.
Importancia:	Vital
Urxencia:	Alta
Estado:	Cumprido
Estabilidade:	Alta

Obxectivo OBX-3	
Nome:	Toma de decisións.
Versión:	1.0

Obxectivo OBX-3(cont)	
Descrición:	Dotar á aplicación coa capacidade de aprendizaxe para a toma de decisións na captura de fotografías en base a eventos e condicións do entorno, de xeito que se procure a maior eficiencia posible, maximizando as mostras de éxito (sendo estas aquelas imaxes nas que se poida detectar algún rostro).
Sub-obxectivos:	Non aplica.
Importancia:	Alta
Urxencia:	Media
Estado:	Cumprido
Estabilidade:	Alta

1.2. Descrición técnica

Tal e como se resume nos obxectivos do apartado anterior, durante este proxecto desenvolveuse unha aplicación para dispositivos móbiles. Concretamente, elixiuse traballar con dispositivos Android fronte a iOS polas vantaxes que este sistema operativo aportaba para o afrontamento do problema (cota de mercado, bibliotecas de código aberto dispoñibles, etc.).

A aplicación é capaz de decidir en que momento tomar unha fotografía co fin de detectar na mesma o rostro do usuario. Para isto, intégranse diferentes algoritmos de aprendizaxe que facilitan a toma de decisións apoiándose nun conxunto de datos de adestramento e a información do entorno (captada mediante os propios sensores do dispositivo e a identificación de determinados comportamentos de usuario e demais eventos disparadores). Procúrase a máxima eficiencia da aplicación `[OBX-3]` mediante a obtención dunha boa porcentaxe de mostras válidas (imaxes de calidade e con rostro, que serven para a construción dun modelo) considerando os limitados recursos que un dispositivo móbil pode ofertar. Compre lembrar que a aplicación está suxeita a requirimentos de tempo real, polo que un factor importante é o consumo. Inténtase porén minimizar o consumo de enerxía e evitar monopolizar os recursos de procesamento.

Para a detección de rostros utilízase un detector facial baixo o método de **Viola-Jones** [7]. Concretamente, empregouse a biblioteca de uso libre de visión artificial **OpenCV**, xa que dispón dunha versión para dispositivos Android.

A algoritmia de aprendizaxe seleccionouse conforme evolucionou o proxecto, tendo presentes en todo momento as limitacións temporais. Ata a data, integráronse dous algoritmos de clasificación supervisada, con dúas etapas diferenciadas (adestramento e explotación). O primeiro deles é ***k-nearest neighbors*** (k-NN) [8], incluído entre os máis simples de todos os algoritmos de aprendizaxe automáti-

co. O segundo é a **máquina de soporte vectorial** (SVM) proporcionada pola biblioteca OpenCV [9].

No referente á información do entorno, foi preciso seleccionar aquelas fontes que realmente fosen de utilidade na toma de decisións. En calquera caso, hai que ter en conta que a dotación de intelixencia non supuxo unha fase illada do proxecto, pois implicou a realización de cambios ao longo da fase de implementación en base á experimentación de alternativas. Por iso, un requisito de deseño é a flexibilidade e modularidade da ferramenta, en canto a que a súa arquitectura permita a incorporación de novas variables de entorno e novos algoritmos de aprendizaxe *a posteriori*.

O dispositivo que se empregou nas fases de desenvolvemento e probas é un **BQ Aquaris E5s**. Este *smartphone* conta coa versión 5.1.1 de Android (Lollipop) que, a pesar de non ser a máis recente (a última é a 7.1.1, Nougat), non presentou ningún problema de compatibilidade no referente ao desenvolvemento da aplicación. Por outro lado, a calidade da imaxe podería chegar a ser un factor limitante hai tres ou catro anos, cando as cámaras frontais dos móbiles non tiñan un rol importante. Non obstante, debido á tendencia actual dos denominados *selfies* e da mellora constante das especificacións dos *smartphones*, a maioría dos dispositivos inclúen unha cámara frontal de 1 ou 2 MP como mínimo, sendo cada vez máis habituais os 5 ou 8 MP en gamas medias. No noso caso, o Aquaris E5s dispón dunha cámara frontal de 5 MP, Full HD a 30 fps.

Finalmente, no relativo ao modelo de traballo, en vistas dun entorno variable, cuns requisitos pouco definidos que foron cambiando no transcurso do proxecto, onde a flexibilidade e a produtividade foron sempre fundamentais, optouse rexeitar os ciclos de vida tradicionais. No seu lugar, empregouse unha metodoloxía áxil propia, baseada no modelo **Scrum** [10]. Isto facilitou a realización de entregas parciais e regulares do produto final, adoptando unha estratexia incremental que permitía o solapamento das diferentes fases do desenvolvemento, en lugar de realizar unha tras outra de xeito illado nun ciclo secuencial ou en cascada.

1.3. Enunciado do alcance do proxecto

A presente sección introduce un conxunto de detalles e compromisos relativos ao proxecto, os cales sentan as bases da envergadura do mesmo. Preténdese así asegurar que todos os *stakeholders* teñen un coñecemento común do seu alcance.

1.3.1. Descrición do alcance

A meta deste proxecto é crear unha aplicación para dispositivos móbiles Android que permita obter un banco de mostras do rostro do usuario. O proceso de mostraxe será levado a cabo con certo criterio. Isto implica que o sistema integrará certa intelixencia artificial orientada á toma de decisións. Para elo, empregaranse un ou máis algoritmos de aprendizaxe supervisado. Este tipo de algoritmos compóñense de dúas etapas: adestramento e explotación. O propio usuario do dispositivo poderá efectuar o adestramento en base aos seus hábitos persoais se o desexa.

A aplicación será empregada como ferramenta de apoio á investigación no Centro Singular de Investigación en Tecnoloxías da Información da Universidade de Santiago de Compostela (CiTIUS). Será utilizada por un grupo reducido de usuarios co fin de obter un banco de mostras de cada un. Estes bancos servirán para elaborar modelos das caras dos individuos, que serán utilizados nun futuro sistema de identificación biométrica. Realizarase un seguimento dos usuarios, de xeito que serán *beta testers* da aplicación, co obxectivo de atopar erros no produto. A ferramenta tamén ofrecerá un amplo abanico de opcións de configuración, co fin de concederlle autoridade ao usuario.

No Capítulo 2 pódese consultar a especificación detallada de funcionalidades e requirimentos do sistema.

1.3.2. Criterios de aceptación do produto

A continuación recóllense os criterios que seguirá o cliente para aceptar o produto:

- A aplicación resulta útil como ferramenta de apoio á investigación, de xeito que se poden empregar os bancos de mostras que xera para crear modelos dos usuarios.
- O produto amosa capacidade de adaptación, permitindo a integración de forma doada de novas fontes de información, eventos e algoritmos de aprendizaxe.
- As mostras obtidas pola aplicación rompen as limitacións actuais. Dispónse de mostras de caras para a aprendizaxe específicas do contexto[1].

[1] A captación de rostros en dispositivos móbiles está suxeita a características específicas tanto a nivel de sensores como relacionadas coa forma de usar o dispositivo, o que esixe que para acadar o mellor rendemento se deba dispor de mostras tomadas no contexto propio do dispositivo e o individuo.

- A ferramenta é compatible cun 65 % dos dispositivos do mercado, tomando como preferencia as versións máis recentes e estendidas de Android.
- Asúmese que o software presenta erros. Sen embargo, co fin de maximizar o número de mostras obtidas, a recuperación da aplicación despois dun fallo ten que ser inmediata. Do mesmo xeito, non se tolera que se produzan fallos cunha frecuencia maior ao 10 % do tempo de uso da aplicación.

Estes criterios impactan directamente na especificación de requirimentos do Capítulo 2.

1.3.3. Entregables

O proxecto terá unha única data de entrega, prevista para o 7 de xullo de 2017, que incluirá:

- Última versión estable da aplicación Android.
- Memoria do proxecto.

Isto non implica que o cliente non vaia poder ter acceso a ningún elemento da configuración do proxecto ata a data de entrega. Ao contrario, seguindo a filosofía Scrum, participará activamente en moitas das actividades contempladas no ciclo de vida, e poderá probar a ferramenta nas sucesivas versións orixinadas ao remate de cada *sprint*.

1.3.4. Exclusións

As funcionalidades da aplicación limitaranse ao especificado no Capítulo 2. Para evitar a máis mínima confusión entre as partes, a continuación déixase constancia de todo aquilo que o produto final non vai contemplar:

- A ferramenta non terá a capacidade para identificar ao usuario do dispositivo, limitarase a tomar mostras de caras, que poderán perfectamente pertencer a un intruso. O filtrado desas mostras non válidas realizarase *a posteriori* pero en ningún caso será tarefa do produto deste proxecto.
- Non todas as mostras tomadas terán que ser válidas para o seu posterior emprego na construción dun modelo do usuario. Asúmese que un pequeno porcentaxe (non más dun 10 %) poderán ter mala calidade (pouca nitidez, demasiado escuras, etc.) ou pertencerán a outro individuo.
- A ferramenta non está orientada a que os usuarios poidan adestrar o teléfono de xeito exhaustivo. Poderase levar a cabo un adestramento propio, pero

non se garante que os resultados superen aos obtidos co adestramento por defecto. O fin desta funcionalidade é puramente demostrativo.

- Aínda que existan criterios de aceptación relativos á compatibilidade, a aplicación non é concibida como un produto que vaia a ser distribuído de xeito masivo, senón que será empregado por un colectivo acoutado.
- Tan só se garantirá a compatibilidade para os dispositivos que cumpran cos requirimentos dispostos no Apéndice C.2 deste documento. En ningún caso a aplicación será compatible con calquera outro sistema operativo para dispositivos móbiles que non sexa Android: Windows Phone, iOS, Ubuntu Phone, etc.

Tamén é importante remarcar que queda fora do alcance do equipo de desenvolvemento calquera tarefa que non sexa a implementación, xestión, despregamento ou mantemento do produto. Ademais, calquera gasto non previsto no plan de custos da Sección 3.5 que encareza o desenvolvemento do proxecto, como poden ser licenzas necesarias para a elaboración da aplicación ou taxas adicionais impostas por provedores de servizos, correrán a conta do cliente, neste caso a USC.

1.3.5. Restricións

O desenvolvemento do proxecto verase limitado polas seguintes restricións:

- O equipo de traballo comporase por un só individuo, que asumirá os roles de Scrum Master, xefe de proxecto, analista, deseñador, programador e encargado de probas (ver plan de xestión de RRHH, Sección 3.4).
- O traballador non poderá adicar a totalidade da súa xornada ao proxecto, senón que de media asignaralle o 40 % do seu tempo, o que aproximadamente se corresponde con 14 horas semanais (ver planificación temporal, Sección 3.3).
- A duración máxima do proxecto non superará as 420 horas de traballo.
- A data límite de entrega será o 7 de xullo.

1.3.6. Supostos

Os supostos do proxecto son os seguintes:

- Empregarase a ferramenta asumindo que a mesma está lonxe de ser un produto rematado.

- Os individuos aos que se lle conceda o acceso á mesma terán un mínimo coñecemento do seu funcionamento: saber diferencias etapas de aprendizaxe, coñecer o significado de cada opción de *settings*, saber localizar as mostras tomadas, etc.

1.4. Participantes

A Táboa 1.4 recolle todas as partes interesadas ou *stakeholders* do proxecto. Para simplificar a táboa, omítese intencionalmente a información sobre a entidade ou organización á que pertence cada individuo posto que todos eles traballan para a Universidade de Santiago de Compostela (USC). Tampouco se proporciona información de contacto do voluntariado por garantir a súa privacidade. Para maior detalle sobre a xestión do persoal, pódense consultar as Seccións 3.4 e 3.7.

Nome	**Contacto**	**Descrición**
Fernando Estévez Casado	`fernando.estevez@rai.usc.es`	Desenvolvedor único de FaceSampler
Xosé M. Pardo López	`xose.pardo@usc.es`	Representante do cliente
Roberto Iglesias Rodríguez	`roberto.iglesias.rodriguez@usc.es`	Cliente
Carlos Vázquez Regueiro	`cvazquez@udc.es`	Cliente
Eric López López	`eric.lopez@usc.es`	Cliente
Voluntariado	-	Conxunto de usuarios que accederon a seren *beta testers* da aplicación.

Táboa 1.4: *Stakeholders* do proxecto.

Capítulo 2

Especificación de requisitos

Tradicionalmente veuse a tratar a captura e a xestión de requisitos do proxecto dun xeito que, cada vez máis, se amosa erróneo. Entendeuse a captura como unha fase temperá, que unha vez se remataba nos daba a fotografía exacta do que necesitaba o cliente. Un problema habitual é o enorme esforzo que supón facer unha captura en detalle dos requisitos. Tan enorme que rara vez xustifica o resultado. Outro dos problemas comúns é que o cliente case nunca ten claras as súas propias necesidades coa profundidade suficiente como para definilas *a priori* e a isto súmase que, acotío, durante a vida do proxecto, ditas necesidades cambian.

En Scrum, os requisitos exprésanse como elementos da **Pila de Produto** ou Product Backlog [10]. A Pila de Produto é unha lista viva ordenada e priorizada dos requisitos funcionais e non funcionais en base ao seu valor para o cliente. Isto implica que os requisitos que nela aparecen e a orde dos mesmos é cambiante ao longo da vida do proxecto. Utilizar a Pila de Produto como principal artefacto para a xestión de requisitos permítenos atallar os problemas relacionados coa xestión de requisitos.

Construír e manter a Pila de Produto é unha tarefa ardua, pero é moito máis sinxelo que facer una captura de requisitos tradicional, posto que tan só hai que expresar a grandes trazos en que consiste cada requisito. Só é preciso o nivel de detalle suficiente para poder estimar os requisitos e priorizalos. Evidentemente, antes da súa implementación será necesario refinar en maior detalle os mesmos, pero en Scrum iso é algo que se pode pospor ata a planificación do *sprint* (iteración) no que vamos a abordar estes requisitos en concreto. Pospor o refinado detallado ata un momento próximo á su implementación permite que cando realicemos esta actividade teñamos en conta de novo as necesidades do cliente que puideron cambiar e contar coa información que adquirimos durante a implementación de outros requisitos. Deste xeito, ao contar con máis información teremos

moitas máis posibilidades de actuar correctamente á hora de describir e estimar os requirimentos.

Un problema que Scrum non resolve é como documentar a captura detallada de requisitos. De feito, non se impón nin suxire ningunha documentación nin ningún circuíto de documentación. O plantexamento en Scrum baséase no principio do Manifesto Áxil [11] que di: "Valorar máis o software que funciona que a documentación exhaustiva". En ocasións, como para a elaboración da presente memoria, a Pila de Produto pode ser insuficiente. Nada impide neste tipo de situacións usar o mecanismo que máis nos conveña para documentar os requisitos: historias de usuario, sesións JAD, casos de uso, escenarios, etc.

Neste caso optouse por realizar a especificación dos requisitos funcionais por medio de **casos de uso**, por tratarse dun método moi estendido e máis que empregado durante a formación no Grao. A maiores, inclúese tamén unha especificación formal dos requisitos de información e non funcionais. Compre ter presente que o que neste capítulo se recolle é unha instantánea da Pila de Produto expresada de xeito formal, realizada expresamente para esta memoria.

2.1. Definición de prioridades

Antes de comezar coa especificación, compre matizar a modo de lenda o significado dos atributos aos cales se recorrerá na definición de cada un dos requisitos.

- **Importancia**:
  - **Quedaría ben**. É un requisito que non foi solicitado polo cliente, pero que sen dúbida aporta valor engadido ao produto final. Soe estar ligado a unha prioridade baixa.
  - **Importante**. É o cliente quen o solicita, de xeito directo ou indirecto (se é algo evidente que, a pesar de non ser comentado polo cliente, se presupón). Non se considera o produto rematado se un requisito marcado con este atributo non foi abordado satisfactoriamente.
  - **Vital**. É un requisito indispensable. Vai asociado á máis alta prioridade e debe ser abordado de inmediato.
- **Urxencia**:
  - **Baixa**. Non é necesario abordalo no momento. Sitúase na parte baixa da Pila de Produto e será traballado nos últimos *sprints* do proxecto.

  - **Moderada**. Non é abordado no momento, pero compre telo suficientemente detallado canto antes.
  - **Alta**. Debe implementarse axiña. Os seus detalles están xa moi pulidos, situándose no máis alto da Pila de Produto. É abordado nun dos primeiros *sprints*.

- **Estado**:
  - **En espera**. O requisito non foi pulido en detalle aínda. Isto implica que existen outros requisitos con maior prioridade por enriba del na Pila de Produto.
  - **En construción**. Estase abarcando nestes momentos. Isto pode implicar que estea sendo analizado para pulir os detalles do mesmo de cara ao próximo *sprint* ou ben que xa estea sendo implementado no *sprint* actual.
  - **Pendente de verificación**. O requisito foi abordado no último *sprint*, pero aínda se debe comprobar se se axusta á súa especificación.
  - **Pendente de validación**. Xa foi verificado, pero aínda hai que comprobar se se axusta ao que o cliente espera.
  - **Validado**. Xa foi verificado e validado, o que implica que a funcionalidade é aceptada.

- **Estabilidade**:
  - **Baixa**. A súa especificación depende de moitas condicións cambiantes, polo que se espera que sexa modificado.
  - **Media**. A súa especificación está ben detallada, pero aínda é cedo para poder consideralo estable. Cumprirá abordalo nas próximas reunións co cliente. Por tanto, é posible que se deban realizar cambios.
  - **Alta**. O requisito está perfectamente definido e o cliente ten claras as ideas respecto do mesmo, polo que é moi improbable que sufra cambios, aínda que sí se contempla a posibilidade de realizar algún cambio menor.

2.2. Requisitos de información

Os seguintes requisitos describen que información debe almacenar, procesar, recuperar e transferir o sistema para poder ofrecer os servizos necesarios. Identifican

conceptos relevantes sobre os que se debe gardar información, así como as propiedades específicas relativas ao ditos conceptos.

Índice

- [RI-1]: Captura.
- [RI-2]: Mostra.
- [RI-3]: Información provinte de sensores.
- [RI-4]: Información provinte de servizos.
- [RI-5]: Cuaternión.
- [RI-6]: Estado.
- [RI-7]: *Dataset* de adestramento.

Catálogo

RI-1 **Captura** (shot)	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	Non aplica.
Descrición:	Toda imaxe tomada polo dispositivo durante o proceso de adestramento ou mostraxe.
Propiedades:	a. Matriz bidimensional de representación da imaxe. b. Número de caras detectadas na imaxe. c. Éxito ou fracaso.
Comentarios:	Recoméndase a lectura da Sección 4.1.

RI-2 **Mostra** (sample)	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[RI-1].
Descrición:	Imaxe axustada da cara do usuario, obtida tras procesar unha captura exitosa.
Propiedades:	Matriz bidimensional de representación da imaxe.
Comentarios:	Sen comentarios.

RI-3 **Información provinte de sensores** (sensorData)	
Versión:	1.0

RI-3 **Información provinte de sensores** (sensorData) *(cont.)*	
Autor:	Fernando Estévez
Dependencias:	Non aplica.
Descrición:	Conxunto de datos tomados do entorno a través dos sensores integrados no dispositivos que serán empregados para a aprendizaxe.
Propiedades:	a. Vector de datos do acelerómetro. b. Vector de datos do xiroscopio. c. Vector de datos do magnetómetro. d. Medición lumínica do sensor de luz.
Comentarios:	A información seleccionada e os sensores que a proporcionan poderán variar nun futuro.

RI-4 **Información provinte de servizos** (serviceData)	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	Non aplica.
Descrición:	Conxunto de datos tomados do entorno a través de servizos Android que realizan un seguimento do usuario.
Propiedades:	Instante de tempo no que se produciu o último toque de pantalla.
Comentarios:	A información seleccionada e os servizos que a proporcionan poderán variar nun futuro.

RI-5 **Cuaternión** (quaternion)	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[RI-3].
Descrición:	Matriz de rotación que representa as transformacións a aplicar para pasar do sistema de referencia local do móbil ao inercial da Terra.
Propiedades:	Vector de datos do cuaternión.
Comentarios:	A información seleccionada e os sensores que a proporcionan poderán variar nun futuro. Para máis detalles, ver Sección 4.1.

RI-6 **Estado** (state)	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[RI-3], [RI-4] e [RI-5].

RI-6 **Estado** (`state`) *(cont.)*	
Descrición:	Vector de dúas compoñentes, patrón e etiqueta, onde o primeiro elemento é o conxunto de variables do entorno empregadas na aprendizaxe do dispositivo e o segundo indica a clase á que pertence ese patrón, que pode ser éxito ou fracaso, en función de se o estado está ligado a unha boa ou mala captura de rostros.
Propiedades:	a. Patrón. b. Etiqueta.
Comentarios:	Ver Sección 4.1.

RI-7 ***Dataset*** **de adestramento** (`model`)	
Versión:	`1.0`
Autor:	Fernando Estévez
Dependencias:	`[RI-6]`.
Descrición:	Conxunto de estados almacenados durante o proceso de adestramento. Este modelo é utilizado para facer as predicións no proceso de mostraxe.
Propiedades:	Conxunto de estados.
Comentarios:	Ver Sección 4.1.

2.3. Requisitos non funcionais

Nesta sección expóñense aquelas condicións que lle foron impostas ao sistema a desenvolver relacionadas con aspectos principalmente de calidade, algúns dos cales inflúen na súa arquitectura. Preséntanse a continuación definidas mediante requisitos non funcionais catalogados por áreas.

2.3.1. Usabilidade

Esta categoría define aspectos relacionados coas dificultades que poden atopar os usuarios ao facer uso da ferramenta. Especifícanse os seguintes requisitos:

- `[RNF-1]`: Informar de procesos en *background*.
- `[RNF-2]`: Notificar os tempos de carga.
- `[RNF-3]`: Indicar a detección dun rostro.
- `[RNF-4]`: Notificar captura.
- `[RNF-5]`: Idiomas.

RNF-1 **Informar de procesos en *background***	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1]
Descrición:	O sistema deberá informar ao usuario en todo momento, de ser o caso, de que se están levando a cabo procesos en segundo plano. O habitual será facelo mediante notificacións fixas á barra de notificacións do sistema.
Importancia:	Quedaría ben
Urxencia:	Baixa
Estado:	Validado
Estabilidade:	Alta
Comentarios:	O obxectivo é cumprir cos criterios de boas prácticas defendidos por Android.

RNF-2 **Notificar os tempos de carga**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1]
Descrición:	O sistema deberá informar ao usuario cun diálogo cando este requira dun servizo ou funcionalidade que implique certo tempo de carga previo. Será necesario que se inclúa este tipo de notificacións cando o tempo de espera sexa igual ou superior a 750 milisegundos.
Importancia:	Quedaría ben
Urxencia:	Baixa
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Sen comentarios.

RNF-3 **Indicar a detección dun rostro**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-2]
Descrición:	Durante o adestramento en primeiro plano, o sistema deberá informar ao usuario de cando se está a recoñecer o seu rostro. Para elo, amosaranse en tempo real as imaxes capturadas co sensor frontal do dispositivo e, cando se de o caso, avisarase da detección da cara enmarcándoa nun cadro de cor rechamante.
Importancia:	Importante
Urxencia:	Moderada
Estado:	Validado
Estabilidade:	Alta

RNF-3 **Indicar a detección dun rostro** *(cont.)*	
Comentarios:	Escóllese por defecto a cor verde clara `#00FF00` para o debuxado do cadro sobre o rostro.

RNF-4 **Notificar captura**	
Versión:	`1.0`
Autores:	Fernando Estévez
Dependencias:	`[OBX-1]`, `[OBX-2]`, `[OBX-3]`
Descrición:	Durante o proceso de mostraxe, cando o sistema decida tomar unha captura, deberá notificala por medio dun aviso textual e sonoro.
Importancia:	Importante
Urxencia:	Moderada
Estado:	Validado
Estabilidade:	Alta
Comentarios:	O aviso sonoro será o habitual *click* empregado nas aplicacións de cámara e poderá ser silenciado se o usuario o desexa.

RNF-5 **Idiomas**	
Versión:	`1.0`
Autores:	Fernando Estévez
Dependencias:	`[OBX-1]`, `[OBX-2]`, `[OBX-3]`
Descrición:	A aplicación estará dotada de traducións ao inglés, español e galego. A selección do idioma será tarefa automática do dispositivo en base á súa configuración lingüística xeral. O idioma por defecto será o inglés no caso de que o usuario non teña o dispositivo configurado nunha destas tres linguas.
Importancia:	Pode esperar
Urxencia:	Baixa
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Sen comentarios.

2.3.2. Seguridade

Esta categoría define aspectos relativos á política de privacidade da aplicación. Posto que esta non inclúe sistemas de control de acceso ou autenticación, o factor destacado aquí é a protección de datos. Identifícanse os seguintes requisitos:

- `[RNF-6]`: Políticas de privacidade.
- `[RNF-7]`: Permisos do sistema.

- [RNF-8]: Control sobre capturas e mostras.

RNF-6 **Políticas de privacidade**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1]
Descrición:	A aplicación contará cunha política de privacidade detallada, accesible para todo usuario da mesma.
Importancia:	Quedaría ben
Urxencia:	Baixa
Estado:	Validado
Estabilidade:	Alta
Comentarios:	O feito de publicar a ferramenta na Google Play Store esixe dispor dunha política de privacidade definida cando se toman permisos delicados do sistema.

RNF-7 **Permisos do sistema**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1]
Descrición:	O sistema deberá solicitar os permisos que necesite (acceso á cámara e almacenamento interno) de xeito que o usuario sexa consciente dos privilexios que lle está outorgando.
Importancia:	Importante
Urxencia:	Moderada
Estado:	Validado
Estabilidade:	Alta
Comentarios:	O xeito de cumprir con este requisito virá dado en función da versión de Android do dispositivo co que traballe o usuario.

RNF-8 **Control sobre capturas e mostras**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1]
Descrición:	Toda captura realizada polo sistema será almacenada de xeito local na memoria interna do dispositivo, incluso cando sexa descartada para poder ser unha mostra do seu rostro. Deste xeito haberá transparencia co usuario, xa que saberá que información se está a recadar e terá o control sobre a mesma, podendo borrar aquelas capturas ou mostras que considere inapropiadas.
Importancia:	Importante
Urxencia:	Moderada

RNF-8 **Control sobre capturas e mostras** *(cont.)*	
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Sen comentarios.

2.3.3. Eficiencia

Esta categoría define aspectos que indican a proporción entre o nivel de cumprimento do software e a cantidade de recursos necesitados baixo condicións establecidas. Obtéñense os seguintes requisitos:

- `[RNF-9]`: Velocidade de adestramento.
- `[RNF-10]`: Velocidade de mostraxe.
- `[RNF-11]`: Velocidade de obtención do banco de mostras.
- `[RNF-12]`: Taxa de acerto na toma de mostras.

RNF-9 **Velocidade de adestramento**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-2], [OBX-3]
Descrición:	O sistema debe ser capaz de procesar 2 imaxes por segundo durante a fase de adestramento en primeiro plano.
Importancia:	Vital
Urxencia:	Alta
Estado:	Validado
Estabilidade:	Alta
Comentarios:	O feito de procesar unha imaxe durante o adestramento implica tamén obter a información do entorno necesaria para o aprendizaxe (o estado).

RNF-10 **Velocidade de mostraxe**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-2], [OBX-3]
Descrición:	O sistema deberá ter un período de mostraxe mínimo de 5 segundos.
Importancia:	Vital
Urxencia:	Alta
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Sen comentarios.

RNF-11 **Velocidade de obtención do banco de mostras**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-2], [OBX-3]
Descrición:	O sistema permitirá que un usuario medio sexa capaz de xerar un banco dun mínimo de 200 mostras en menos de 15 días de uso continuado.
Importancia:	Vital
Urxencia:	Alta
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Sen comentarios.

RNF-12 **Taxa de acerto na toma de mostras**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-3]
Descrición:	O sistema deberá tomar capturas con certo criterio, de xeito que se permita que, de media, polo menos un 40 % desas capturas se convertan en mostras válidas.
Importancia:	Vital
Urxencia:	Alta
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Sen comentarios.

2.3.4. Portabilidade e escalabilidade

Esta categoría define aspectos relacionados coa capacidade do sistema para adaptarse aos cambios de requirimentos, así como aos distintos dispositivos e plataformas. Os requisitos non funcionais asociados son:

- [RNF-13]: Escalabilidade do módulo de aprendizaxe.
- [RNF-14]: Compatibilidade software.
- [RNF-15]: Compatibilidade hardware.

RNF-13 **Escalabilidade do módulo de aprendizaxe**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-3]
Descrición:	O sistema debe ser capaz de incorporar novos algoritmos de aprendizaxe.

RNF-13 **Escalabilidade do módulo de aprendizaxe** *(cont.)*	
Importancia:	Vital
Urxencia:	Alta
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Sen comentarios.

RNF-14 **Compatibilidade software**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1]
Descrición:	A aplicación debe ser como mínimo compatible coas versións máis recentes e extendidas de Android. Considérase imprescindible que poida ser instalada en dispositivos con Android 5.0 (Lollipop) ou superior.
Importancia:	Importante
Urxencia:	Moderada
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Sen comentarios.

RNF-15 **Compatibilidade hardware**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1]
Descrición:	A aplicación deberá funcionar con normalidade en todo aquel dispositivo que conte cos sensores mínimos indispensables para a captura e o aprendizaxe: sensor de cámara interior, acelerómetro e xiroscopio.
Importancia:	Importante
Urxencia:	Moderada
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Sen comentarios.

2.3.5. Fiabilidade

Esta categoría define aspectos relacionados coa capacidade do produto para manter o seu nivel de prestación baixo condicións definidas e durante un período de tempo establecido. Identifícase un único requisito non funcional:

RNF-16 **Dispoñibilidade**	
Versión:	1.0
Autores:	Fernando Estévez
Dependencias:	[OBX-1]
Descrición:	Asúmese a inestabilidade no produto, pero a probabilidade de falla nun día completo de uso da ferramenta non poderá exceder o 10 %.
Importancia:	Importante
Urxencia:	Moderada
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Sen comentarios.

2.4. Catálogo de casos de uso

A continuación preséntase o catálogo completo de casos de uso da aplicación. A Figura 2.1 amosa un diagrama en UML cos mesmos, agrupados en tres categorías: **navegación**, **control de procesos** e **xestión de preferencias**.

Nesta aplicación non se contemplan roles de usuario. Isto quere dicir que todo individuo que a empregue terá os mesmos privilexios. Porén, o noso sistema contempla **un único actor**:

Usuario Todo individuo que instale a aplicación e faga uso normal da mesma no seu dispositivo móbil Android.

2.4.1. Navegación

Inclúense nesta categoría aqueles casos de uso relacionados coa interacción entre o usuario e as distintas vistas da aplicación, que non teñen relación directa coas funcionalidades principais que brinda a mesma:

- [CU-1]: Iniciar a aplicación.
- [CU-2]: Permutar entre mostraxe e adestramento.
- [CU-3]: Acceder ao panel de preferencias.
- [CU-4]: Volver á vista anterior.
- [CU-5]: Pausar a aplicación.
- [CU-6]: Retomar a aplicación.

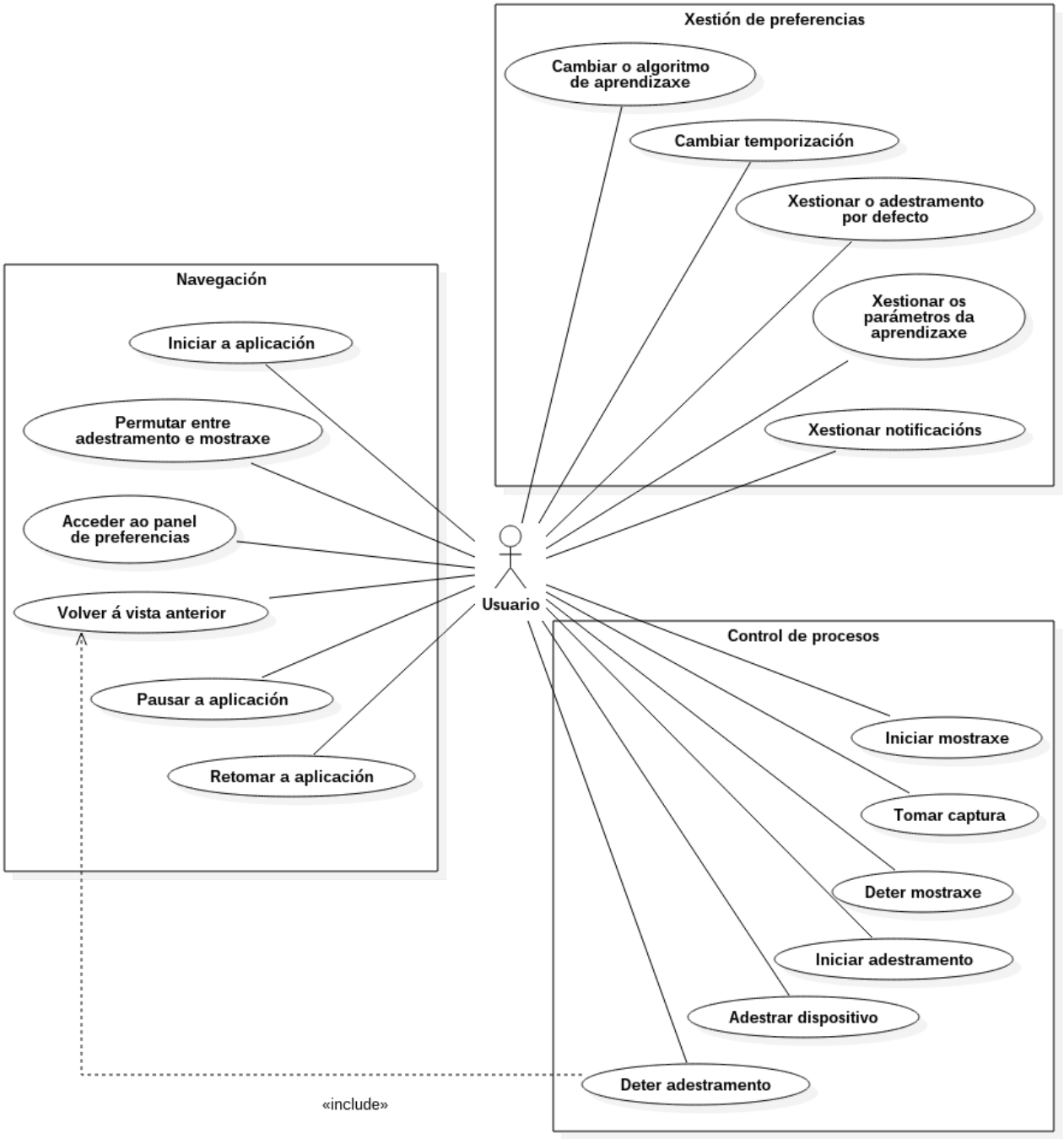


Figura 2.1: Diagrama global de casos de uso.

CU-1 **Iniciar a aplicación**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1]
Descrición:	O usuario identifica a aplicación no menú de aplicacións do seu dispositivo persoal e iníciaa.
Actor Principal:	Usuario
Precondicións:	A aplicación foi instalada exitosamente no dispositivo en cuestión.
Postcondicións:	Ao usuario amosaráselle a vista principal da aplicación.

Escenario principal:

1. O usuario accede ao menú de aplicacións do dispositivo.
2. O usuario identifica FaceSampler en base á súa icona e/ou o seu nome.
3. O usuario preme co dedo na icona da ferramenta e esta é lanzada.
4. Durante o lanzamento da aplicación, o sistema avalía se o dispositivo cumpre coas dependencias necesarias para o correcto funcionamento da mesma.
5. O usuario ve a vista principal da aplicación, que será aquela que por defecto permita iniciar a mostraxe, posto que é a finalidade principal da mesma.

Extensións:

5.a Se por calquera motivo o usuario non conta coa biblioteca OpenCV instalada no dispositivo, non poderá empregar a ferramenta con normalidade. Porén, ao iniciarse a aplicación amosarase un diálogo que lle facilitará o acceso á descarga da biblioteca desde a Play Store de Google. Este diálogo non poderá pecharse ata que o usuario resolva o problema, impedíndolle así o acceso ás demais funcionalidades da ferramenta.

5.b Se o dispositivo do usuario carece dalgún sensor que prové ao sistema de información vital para o aprendizaxe, entón notificarase o problema por pantalla cun diálogo só a primeira vez que se inicie a aplicación. O sistema será quen de resolver esta carencia e funcionar igualmente, aínda que sexa de xeito menos efectivo ou eficiente.

Importancia:	Vital
Urxencia:	Alta
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Non se impuxeron restricións en canto ao nome da aplicación nin o deseño da súa icona. Propúxose FaceSampler como título e o actual logo como icona. Ambos foron aceptados polo cliente.

`CU-1` **Iniciar a aplicación** *(cont.)*	
Proba de aceptación:	O cliente poderá probar a iniciar a aplicación de xeito habitual. A maiores, simularanse as situacións nas que se careza de OpenCV e algún dos sensores necesarios, de xeito que el poida ver como se resolven estes inconvenientes.

`CU-2` **Permutar entre mostraxe e adestramento**	
Versión:	`1.0`
Autor:	Fernando Estévez
Dependencias:	`[OBX-1]`, `[CU-1]`
Descrición:	O usuario cambia entre os modos de adestramento e explotación da aplicación.
Actor Principal:	Usuario
Precondicións:	O usuario iniciou a aplicación `[CU-1]` seguindo o escenario principal.
Postcondicións:	A interface gráfica reflectirá sen lugar a confusión a selección do usuario.
Escenario principal:	1. O usuario, situado na vista principal da aplicación, en modo mostraxe, cambia ao modo adestramento deslizando co dedo de esquerda a dereita sobre a pantalla. 2. A interface gráfica actualízase conforme á temática de adestramento.
Extensións:	1.a Se o usuario xa houbese cambiado antes de modo, situándose agora na vista de adestramento, o proceso será inverso, tendo neste caso que deslizar de dereita a esquerda.
Importancia:	Quedaría ben
Urxencia:	Baixa
Estado:	Validado
Estabilidade:	Alta
Comentarios:	A pesar de que o cliente non solicitou esta funcionalidade, considerouse que aporta valor ao produto mellorando a súa usabilidade, pois non se pode comparar con ter todo o control de adestramento e mostraxe mesturado nunha vista simple. Nas cuestións de estética e de usabilidade o cliente sempre deu liberdade.
Proba de aceptación:	Deixarase que o cliente permute entre modos e pedirase a súa aprobación da usabilidade da interface.

CU-3 **Acceder ao panel de preferencias**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [CU-1]
Descrición:	O usuario entra no panel de preferencias da aplicación.
Actor Principal:	Usuario
Precondicións:	O usuario iniciou a aplicación [CU-1] seguindo o escenario principal.
Postcondicións:	Amósase o panel de preferencias actualizado, permitindo axustar as preferencias dispoñibles no momento, xa que non todas poden ser modificadas en determinados estados do sistema (durante a mostraxe, en modo aforro de enerxía, etc).
Escenario principal: 1. O usuario, situado na vista principal da aplicación (ben sexa en modo adestramento ou mostraxe), accede ao panel de axustes pulsando na icona en forma de engrenaxe situada na esquina superior dereita da pantalla.	
Extensións:	Non aplica
Importancia:	Importante
Urxencia:	Moderada
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Se algunha preferencia non pode ser modificada no momento non ten por que ocultarse, poderá ser bloqueada e mostrarse nunha cor escurecida para que o usuario saiba que non pode manipulala.
Proba de aceptación:	O cliente poderá acceder á pantalla de preferencias e estas amosaranse actualizadas en función do estado do sistema.

CU-4 **Volver á vista anterior**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [CU-1], [CU-3]
Descrición:	O usuario retrocede de xeito voluntario á vista inmediatamente precedente.
Actor Principal:	Usuario
Precondicións:	O usuario iniciou a aplicación e, partindo da vista principal, accedeu a unha vista secundaria, ben sexa o panel de preferencias, a vista de adestramento en primeiro plano, etc.

CU-4 **Volver á vista anterior** *(cont.)*	
Postcondicións:	Amósase a vista anterior (polo xeral a vista principal da aplicación), mantendo a coherencia co estado que presentaba antes de abandonala.
Escenario principal: 1. O usuario, situado nunha vista secundaria da aplicación, decide volver á vista inmediatamente superior. Para iso, preme na icona de retroceso, ubicada na esquina superior esquerda da pantalla e caracterizada por tratarse dunha frecha apuntando cara á esquerda (←). 2. O sistema, que leva constancia do historial de navegación, recupera a vista oportuna.	
Extensións: 1.a O usuario preme no botón físico de retroceso do seu dispositivo, en caso de que este exista.	
Importancia:	Importante
Urxencia:	Moderada
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Non aplica
Proba de aceptación:	O cliente poderá retornar á vista principal tanto desde a vista de preferencias como desde a de adestramento en primeiro plano.

CU-5 **Pausar a aplicación**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [CU-1]
Descrición:	O usuario envía a aplicación a segundo plano para acceder á pantalla de inicio do dispositivo.
Actor Principal:	Usuario
Precondicións:	O usuario iniciou a aplicación.
Postcondicións:	Non aplica.
Escenario principal: 1. O usuario, situado en calquera das vistas da aplicación, decide pausar a ferramenta. Para iso, preme no botón físico *home* do seu dispositivo. 2. O sistema durme os procesos relacionados coa interface gráfica. Aqueles procesos alleos á vista non son detidos, senón que se comportan en función das súas especificacións propias (ver Sección 4.2).	

`CU-5` **Pausar a aplicación** *(cont.)*	
Extensións: 1.a O usuario preme no botón físico de bloqueo do seu dispositivo, durmindo así ao mesmo.	
Importancia:	Vital
Urxencia:	Alta
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Non aplica
Proba de aceptación:	O cliente poderá pausar a aplicación desde calquera das vistas da mesma, accedendo así á pantalla de inicio.

`CU-6` **Retomar a aplicación**	
Versión:	`1.0`
Autor:	Fernando Estévez
Dependencias:	`[OBX-1]`, `[CU-1]`, `[CU-5]`
Descrición:	O usuario trae a aplicación a primeiro plano.
Actor Principal:	Usuario
Precondicións:	O usuario iniciou a aplicación e posteriormente a pausou.
Postcondicións:	Amósase a aplicación no estado exacto no que quedou no momento de ser pausada.
Escenario principal: 1. O usuario, situado no menú de aplicacións do dispositivo, preme na icona de FaceSampler para retomar a súa execución. 2. O sistema desperta os procesos relacionados coa interface gráfica, retomándoos no estado exacto no que quedaron antes da pausa. De haber outros procesos en execución, relacionados coa captura de mostras ou a aprendizaxe, son xestionados en función das súas especificacións propias (ver Sección 4.2).	
Extensións: 1.a O usuario retoma a aplicación desbloqueando o dispositivo. Só se aplica cando este foi bloqueado estando FaceSampler en primeiro plano (extensión 1.a `[CU-5]`). 1.b O usuario retoma a aplicación desde o xestor de aplicacións en execución do dispositivo.	
Importancia:	Vital
Urxencia:	Alta
Estado:	Validado

CU-6 **Retomar a aplicación** *(cont.)*	
Estabilidade:	Alta
Comentarios:	Cando se di que o estado da aplicación ten que ser o mesmo antes e despois da pausa e recomezo, preténdese englobar todas aquelas propiedades relacionadas coa interface de usuario: vista amosada, historial de navegación, etc. En xeral, considerarase que se mantén o mesmo estado cando a aplicación se comporta como se nunca se houbese producido unha pausa. Posto que algunhas vistas poden ser dinámicas, se algún proceso independente á interface gráfica se segue executando en segundo plano durante a pausa e os seus resultados son amosados por pantalla, dita vista deberá estar actualizada no momento do recomezo.
Proba de aceptación:	O cliente poderá retomar a aplicación no momento que el desexe, recuperándoa no estado exacto no que a pausou.

2.4.2. Control de procesos

Esta categoría comprende os casos de uso relativos ao control dos grandes procesos do sistema, adestramento e mostraxe, ambos ligados á aprendizaxe do mesmo:

- [CU-7]: Iniciar mostraxe.
- [CU-8]: Tomar captura.
- [CU-9]: Deter mostraxe.
- [CU-10]: Iniciar adestramento.
- [CU-11]: Adestrar dispositivo.
- [CU-12]: Deter adestramento.

CU-7 **Iniciar mostraxe**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-2], [OBX-3], [CU-1], [RNF-1]
Descrición:	O usuario ordena ao sistema que comece o proceso de mostraxe.
Actor Principal:	Usuario
Precondicións:	O usuario iniciou a aplicación [CU-1] seguindo o escenario principal.

CU-7 **Iniciar mostraxe** *(cont.)*	
Postcondicións:	O usuario será consciente de que se iniciou a mostraxe grazas ás notificacións visuais.
Escenario principal:	1. Desde a vista principal da aplicación, modo mostraxe, o usuario preme no botón que permite comezar o proceso. 2. O sistema notifica o comezo do mesmo cunha mensaxe en pantalla. Do mesmo xeito, queda fixada á barra de notificacións unha notificación que permite ao usuario saber en todo momento que o proceso está en execución [RNF-1]. 3. O usuario poderá agora realizar un uso habitual do dispositivo, pechando a aplicación, mandándoa a segundo plano, bloqueando o móbil, etc. O proceso nunca finalizará ata que el non o ordene, e comportarase en base a un conxunto de regras lóxicas detalladas na Sección 4.2.2.
Extensións:	Non aplica.
Importancia:	Vital
Urxencia:	Alta
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Non aplica.
Proba de aceptación:	O cliente poderá iniciar a mostraxe desde a vista principal. Será notificado como se describe no escenario principal. Poderá deixar o proceso en execución durante o tempo que el desexe: minutos, horas ou días para avaliar o rendemento da mostraxe, pero o proceso nunca se deterá por conta propia.

CU-8 **Tomar captura**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-2], [OBX-3], [CU-1], [RNF-1], [RNF-4]
Descrición:	O sistema realiza unha captura do usuario.
Actor Principal:	Usuario
Precondicións:	O usuario iniciou o proceso de mostraxe.
Postcondicións:	O usuario poderá ver a captura realizada.
Escenario principal:	

CU-8 **Tomar captura** *(cont.)*	
1. O usuario está usando o dispositivo mentres a mostraxe se executa en segundo plano. 2. O sistema cre que se dan as condicións idóneas para obter unha mostra do usuario, co cal realiza unha captura coa cámara frontal do dispositivo, gardándoa na galería de capturas. 3. O usuario é notificado [RNF-4]. 4. O sistema avalía a captura e decide se serve como mostra. De ser así, gárdase na galería de mostras. 5. O sistema prosegue a mostraxe seguindo o conxunto de regras lóxicas definidas na Sección 4.2.2.	
Extensións:	Non aplica.
Importancia:	Vital
Urxencia:	Alta
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Non aplica.
Proba de aceptación:	O cliente poderá forzar as condicións para que o sistema decida tomar unha captura (empregando o algoritmo *always true*, ver Sección 4.1). Enterarase de cando esta é realizada e poderá vela na galería de capturas ao momento. De valer para mostra, tamén poderá ver a mesma na galería de mostras.

CU-9 **Deter mostraxe**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [CU-1], [CU-7], [RNF-1]
Descrición:	O usuario ordena ao sistema que remate o proceso de mostraxe.
Actor Principal:	Usuario
Precondicións:	O usuario iniciou o proceso de mostraxe.
Postcondicións:	O usuario será consciente de que rematou o proceso grazas ás notificacións visuais.
Escenario principal:	

CU-9 **Deter mostraxe** *(cont.)*	
1. Desde a vista principal da aplicación, modo mostraxe, o usuario preme no botón que permite deter o proceso. 2. O sistema notifica o fin do mesmo cunha mensaxe en pantalla. Do mesmo xeito, desaparece a notificación que estaba fixada na barra de notificacións [RNF-1]. 3. O usuario prosegue co uso normal da aplicación e do dispositivo.	
Extensións: 1.a O usuario preme no botón de parada integrado na propia notificación do proceso en execución.	
Importancia:	Vital
Urxencia:	Alta
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Non aplica.
Proba de aceptación:	O cliente poderá deter a mostraxe desde a vista principal e desde a notificación. Asegurarase de que o proceso realmente se detivo observando o xestor de tarefas en execución. Comprobará que desde a vista principal da aplicación se permite iniciar unha nova mostraxe.

CU-10 **Iniciar adestramento**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [CU-1], [RNF-2]
Descrición:	O usuario ordena ao sistema que comece o proceso de adestramento.
Actor Principal:	Usuario
Precondicións:	O usuario iniciou a aplicación [CU-1] seguindo o escenario principal.
Postcondicións:	Amosarase a vista de adestramento en primeiro plano.
Escenario principal: 1. Desde a vista principal da aplicación, modo adestramento, o usuario preme no botón que permite comezar o proceso. 2. O sistema solicita os recursos necesarios para o adestramento e lanza a vista de adestramento en primeiro plano.	
Extensións:	Non aplica.
Importancia:	Importante

`CU-10` **Iniciar adestramento** *(cont.)*	
Urxencia:	Moderada
Estado:	Validado
Estabilidade:	Alta
Comentarios:	No seguinte caso `[CU-11]` especifícanse as características da citada vista de aprendizaxe.
Proba de aceptación:	O cliente poderá iniciar o adestramento desde a vista principal.

`CU-11` **Adestrar dispositivo**	
Versión:	`1.0`
Autor:	Fernando Estévez
Dependencias:	`[OBX-1]`, `[OBX-2]`, `[OBX-3]`, `[CU-1]`, `[CU-10]`, `[RNF-3]`
Descrición:	O usuario instrúe ao dispositivo, de xeito que este logo saiba elixir o momento oportuno para realizar capturas co fin de obter mostras válidas.
Actor Principal:	Usuario
Precondicións:	O usuario iniciou o adestramento.
Postcondicións:	Non aplica.
Escenario principal:	1. O usuario iniciou o proceso, polo que se atopa na vista de adestramento en primeiro plano. 2. O usuario vese retratado en tempo real. O sistema proporciona, información continua dos parámetros empregados na aprendizaxe. 3. Cando se detecta o rostro do usuario, o sistema notifica ao mesmo `[RNF-3]`.
Extensións:	Non aplica.
Importancia:	Importante
Urxencia:	Moderada
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Polo de agora, de cara a unha demostración, é dabondo con permitir un adestramento en primeiro plano. Sen embargo, nun futuro, poderíase integrar o proceso en segundo plano, tal e como se fai na mostraxe.
Proba de aceptación:	O cliente poderá instruír ao seu dispositivo durante o tempo que desexe, comprobando que os parámetros amosados por pantalla son correctos e se notifica a detección do rostro. Posteriormente comprobarase que o coñecemento adquirido se ve reflectido no proceso de mostraxe.

CU-12 **Deter adestramento**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-3], [CU-1], [CU-4], [CU-10]
Descrición:	O usuario detén o proceso de adestramento.
Actor Principal:	Usuario
Precondicións:	O usuario iniciou o proceso de adestramento.
Postcondicións:	Amosarase a vista principal da aplicación, no modo adestramento.
Escenario principal:	1. Desde a vista de adestramento o usuario retrocede á vista principal [CU-4]. 2. O sistema actualiza o banco de datos do aprendizaxe co novo modelo xerado.
Extensións:	Non aplica.
Importancia:	Importante
Urxencia:	Moderada
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Tras a finalización dun novo adestramento, o sistema esquecerá, de existir, o adestramento previo realizado polo usuario.
Proba de aceptación:	O cliente finalizará o adestramento premendo no botón de regreso (ben o físico ou o proporcionado pola aplicación). Poderá comprobar que se actualizou o banco de datos accedendo aos arquivos da aplicación.

2.4.3. Xestión de preferencias

Nesta categoría inclúense todos os casos de uso relacionados coa xestión de preferencias da aplicación, que afectan de forma directa á experiencia de usuario e ao funcionamento dos procesos principais (mostraxe e adestramento):

- [CU-13]: Cambiar o algoritmo de aprendizaxe.
- [CU-14]: Cambiar temporización.
- [CU-15]: Xestionar o adestramento por defecto.
- [CU-16]: Xestionar os factores da aprendizaxe.
- [CU-17]: Xestionar notificacións.

CU-13 **Cambiar o algoritmo de aprendizaxe**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-3], [CU-1], [CU-3], [CU-7]
Descrición:	O usuario cambia o algoritmo de aprendizaxe.
Actor Principal:	Usuario
Precondicións:	O usuario navegou ata o panel de preferencias.
Postcondicións:	Actualizarase a información do panel de preferencias conforme á selección.
Escenario principal: 1. Desde o panel de preferencias, o usuario toca sobre a opción de algoritmo de aprendizaxe. 2. Aparece un diálogo onde se amosan todas as alternativas, coa selección actual marcada. 3. O usuario escolle outro algoritmo e os efectos son aplicados de inmediato no funcionamento do sistema.	
Extensións:	Non aplica.
Importancia:	Quedaría ben
Urxencia:	Baixa
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Se a mostraxe está en execución, a opción aparece deshabilitada posto que os cambios non se poderían aplicar ata o remate do proceso.
Proba de aceptación:	O cliente escolle outro algoritmo. Para comprobar que realmente se efectuou o cambio, poderá ver o *log* do sistema desde o IDE Android Studio co fin de comparar as trazas xeradas cun algoritmo e con outro durante o proceso de mostraxe.

CU-14 **Cambiar temporización**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-2], [OBX-3], [CU-1], [CU-3]
Descrición:	O usuario cambia a temporización do sistema.
Actor Principal:	Usuario
Precondicións:	O usuario navegou ata o panel de preferencias.
Postcondicións:	Actualizarase a información do panel de preferencias conforme á selección.
Escenario principal:	

`CU-14` **Cambiar temporización** *(cont.)*	
1. Desde o panel de preferencias, o usuario toca sobre unha opción da categoría "Temporización" (*Timings*). 2. Aparece un diálogo cun selector numérico (*number picker*) da magnitude apropiada (milisegundos, segundos, minutos, etc). 3. O usuario escolle outra cantidade e os efectos son aplicados de inmediato na velocidade do sistema, se é que algún proceso se está executando.	
Extensións:	Non aplica.
Importancia:	Quedaría ben
Urxencia:	Baixa
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Actualmente permítense catro modificacións temporais que afectan á velocidade dos procesos de adestramento e mostraxe (ver Sección 4.2.2)
Proba de aceptación:	O cliente proba a cambiar cada un dos períodos posibles. Para comprobar que realmente se efectúan os cambios, poderá ver o *log* do sistema desde Android Studio co fin de ver as trazas de cada proceso acompañadas do momento no que se efectúa cada acción.

`CU-15` **Xestionar o adestramento por defecto**	
Versión:	`1.0`
Autor:	Fernando Estévez
Dependencias:	`[OBX-1]`, `[OBX-3]`, `[CU-1]`, `[CU-3]`, `[CU-12]`
Descrición:	O usuario cambia o *dataset* a empregar na mostraxe, pudendo escoller entre un por defecto ou un creado por el no último adestramento realizado. A aplicación instalarase coa opción de adestramento por defecto marcada, e non poderá cambiarse ata que o usuario realice un.
Actor Principal:	Usuario
Precondicións:	O usuario realizou un adestramento propio e logo navegou ata o panel de preferencias.
Postcondicións:	Actualizarase o panel conforme á selección.
Escenario principal: 1. Desde o panel de preferencias, o usuario toca sobre o cadro de verificación (*checkbox*) chamado "Adestramento por defecto" (*Default training*). 2. Se a opción estaba marcada, desmarcarase, e ao revés. Os efectos son aplicados na seguinte mostraxe.	

CU-15 **Xestionar o adestramento por defecto** *(cont.)*	
Extensións:	Non aplica.
Importancia:	Quedaría ben
Urxencia:	Baixa
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Se a mostraxe está en execución, a opción aparece deshabilitada posto que os cambios non se poderían aplicar ata o remate do proceso.
Proba de aceptación:	O cliente proba a cambiar a configuración. Para comprobar que realmente se efectúan os cambios, poderá ver de ver a traza da mostraxe onde se indica se se está a empregar o adestramento por defecto desde o IDE.

CU-16 **Xestionar os factores da aprendizaxe**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [OBX-3], [CU-1], [CU-3], [CU-7]
Descrición:	O usuario activa e desactiva os distintos parámetros empregados no aprendizaxe.
Actor Principal:	Usuario
Precondicións:	O usuario navegou ata o panel de preferencias.
Postcondicións:	Actualizarase a información do panel de preferencias conforme á selección.
Escenario principal:	1. Desde o panel de preferencias, o usuario toca sobre un elemento da categoría "Factores de aprendizaxe" (*Learning factors*). 2. Se dito elemento estaba activo pasará a estar inactivo, e ao revés. Cada elemento é, porén un interruptor (*switch*). Os efectos son aplicados de inmediato no funcionamento do sistema.
Extensións:	Non aplica.
Importancia:	Quedaría ben
Urxencia:	Baixa
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Se a mostraxe está en execución, a opción aparece deshabilitada posto que os cambios non se poderían aplicar ata o remate do proceso. Debido a que os factores poderán cambiar nun futuro, este panel tamén se verá modificado de ser necesario. Non teñen por que proporcionarse axustes para todos os elementos, tan só os que se considere oportuno permitir modificar.

CU-16 **Xestionar os factores da aprendizaxe** *(cont.)*	
Proba de aceptación:	O cliente proba a cambiar algún elemento. Para comprobar que realmente se efectúan os cambios, poderá ver o *log* do sistema desde Android Studio co fin de ver a traza da mostraxe onde se indica os parámetros empregados.

CU-17 **Xestionar notificacións**	
Versión:	1.0
Autor:	Fernando Estévez
Dependencias:	[OBX-1], [CU-1], [CU-3]
Descrición:	O usuario activa e desactiva as distintas notificacións dispostas na aplicación, ben sexan visuais, sonoras, mediante vibración, etc.
Actor Principal:	Usuario
Precondicións:	O usuario navegou ata o panel de preferencias.
Postcondicións:	Actualizarase a información do panel de preferencias conforme aos cambios realizados.
Escenario principal:	1. Desde o panel de preferencias, o usuario toca sobre un elemento da categoría "Notificacións" (*Notifications*). 2. Se a opción estaba marcada, desmarcarase, e ao revés. Os efectos son aplicados de inmediato no funcionamento do sistema.
Extensións:	Non aplica.
Importancia:	Quedaría ben
Urxencia:	Baixa
Estado:	Validado
Estabilidade:	Alta
Comentarios:	Non se ten por que permitir a xestión de todas as notificacións do sistema. Por exemplo, a notificación de execución dun servizo en segundo plano non poderá ser deshabilitada, seguindo os criterios de boas prácticas de Android.
Proba de aceptación:	O cliente proba a cambiar algún elemento. Para verificar que realmente se efectúan os cambios, poderase forzar a execución do evento que lanza a notificación en cuestión.

2.4.4. Matrices de trazabilidade

Para rematar o catálogo de casos de uso, preséntanse a continuación dúas matrices de trazabilidade. A primeira delas, Táboa 2.1, amosa as relacións entre os diversos casos de uso e os tres obxectivos do proxecto, permitíndonos avaliar o valor de cada CU e así priorizalos correctamente. Unha fila baleira nesta matriz implicaría un caso de uso innecesario, xa que non ten dependencias con ningún obxectivo do proxecto. A segunda matriz, Táboa 2.2, reflicte as dependencias entre os propios casos de uso (omítense columnas baleiras para facilitar a súa observación).

	[OBX-1]	[OBX-2]	[OBX-3]
[CU-1]	⇑		
[CU-2]	⇑		
[CU-3]	⇑		
[CU-4]	⇑		
[CU-5]	⇑		
[CU-6]	⇑		
[CU-7]	⇑	⇑	⇑
[CU-8]	⇑	⇑	⇑
[CU-9]	⇑		
[CU-10]	⇑		
[CU-11]	⇑	⇑	⇑
[CU-12]	⇑		⇑
[CU-13]	⇑		⇑
[CU-14]	⇑	⇑	⇑
[CU-15]	⇑		⇑
[CU-16]	⇑		⇑
[CU-17]	⇑		

Táboa 2.1: Matriz de trazabilidade CU-OBX.

	[CU-1]	[CU-3]	[CU-4]	[CU-5]	[CU-7]	[CU-10]	[CU-12]
[CU-1]							
[CU-2]	⇑						
[CU-3]	⇑						
[CU-4]	⇑	⇑		⇑			
[CU-5]	⇑						
[CU-6]	⇑						
[CU-7]	⇑						
[CU-8]	⇑				⇑		
[CU-9]	⇑				⇑		
[CU-10]	⇑						
[CU-11]	⇑					⇑	
[CU-12]	⇑		⇑			⇑	
[CU-13]	⇑	⇑			⇑		
[CU-14]	⇑	⇑					
[CU-15]	⇑	⇑					⇑
[CU-16]	⇑	⇑			⇑		
[CU-17]	⇑	⇑					

Táboa 2.2: Matriz de trazabilidade CU-CU.

Capítulo 3

Xestión do proxecto

A xestión de proxectos é unha das fontes clave no proceso de desenvolvemento de software. De xeito breve, poderíase resumir como o conxunto de técnicas e tarefas para lograr conducir o proxecto desde o seu comezo ata un final satisfactorio. Tradicionalmente, as metodoloxías de xestión de proxectos como PMBOK [12] tiveron unha forte orientación preditiva. É dicir, a partir do detalle do produto que se quere elaborar (análise funcional e técnico, requirimentos, etc.), defínense fases e actividades perfectamente planificadas no tempo en base aos recursos dispoñibles. Partindo desta proxección inicial, o obxectivo durante o transcurso do proxecto é conseguir que se cumpra aquelo que se previra no inicio: calendario, custos e calidade.

Este tipo de metodoloxías poder resultar útil, mellorando a calidade e reducindo as desviacións nos proxectos nos que son aplicadas. Sen embargo, poden presentar determinados inconvenientes, sendo o máis destacado no noso caso a **incerteza**. Este proxecto situouse nun entorno rápido e inestable, onde cumprir o plan inicial non sería garantía de éxito. Desde o seu inicio previánse cambios nas súas necesidades e prioridades, e sería fundamental ter capacidade de adaptación a partir da retroalimentación e incorporación de novas ideas.

En definitiva, a creación de valor mediante a adaptación ás necesidades cambiantes aparece nun primeiro plano fronte á tradicional idea de deseñar un plan e cumprir uns calendarios e requirimentos estáticos. A **xestión áxil de proxectos** permite iniciar o desenvolvemento sen un detalle pechado do que vai a ser construído. Neste caso elixiuse **Scrum** [10] como estratexia áxil para a xestión e desenvolvemento da aplicación. Máis que unha metodoloxía, Scrum é un **marco de traballo**. Isto quere dicir que non define exactamente o que se debe facer, senón que proporciona un conxunto de boas prácticas a seguir para asegurar o éxito do proxecto.

Unha das características máis destacadas de Scrum é o seu enfoque orientado a

maximizar o rendemento do equipo de traballo, sen embargo, neste proxecto o equipo é unipersoal, sendo moitas das pautas propostas moi difíciles de aplicar. Porén, compre remarcar que a estratexia seguida na xestión e o desenvolvemento do proxecto é unha **adaptación** do marco Scrum ás restricións específicas deste proxecto. Nas vindeiras seccións recóllese a posta en práctica das peculiaridades da filosofía Scrum ao longo da vida do mesmo.

3.1. Plan de xestión da configuración

Nesta sección exponse o plan de xestión da configuración do proxecto. Este plan permite dispor dun conxunto de actividades deseñadas para identificar e definir os elementos do sistema que probablemente cambien, controlando o cambio dos mesmos ao longo do seu ciclo de vida, establecendo relacións entre eles, definindo mecanismos para xestionar distintas versións destes elementos e auditando e informando dos cambios realizados.

Propósito e alcance

O propósito deste plan é o de establecer e manter a integridade dos produtos resultado do proxecto a través do ciclo de vida do proceso de software. Seguindo a filosofía Scrum, evitarase a toda costa que a xestión da configuración deteña o avance do proxecto. Para elo defínese unha estratexia de xestión lixeira, pero ao mesmo tempo efectiva e suficiente para o contexto do proxecto: aplicación Android na que traballa un só desenvolvedor.

Elementos de configuración

Identifícanse os seguintes elementos de configuración do software (ECS) como obxecto das técnicas de xestión da configuración:

- **Código fonte**. Proxecto Android Studio con todo o código fonte da aplicación.
- **Memoria do proxecto**. O presente documento.
- **Elementos do deseño**. Conxunto de diagramas UML e demais figuras e *mockups* relativos ao deseño do software.
- **Elementos de xestión**. Conxunto de documentos relativos á xestión do proxecto, como poden ser cronograma, catálogo de requisitos, informes e patróns, etc.

Metodoloxía e ferramentas

Séguese unha metodoloxía propia que respecta o modelo proposto por Scrum. No caso do código fonte e da memoria do proxecto, requírese dun sistema de control que permita recuperar calquera versión anterior destes elementos en calquera instante de tempo. Para isto, faise uso do sistema de control de versións distribuído **Git** [13], asociado a dous repositorios remotos en **GitLab** (`https://gitlab.com`), un para o código fonte e outro para a memoria.

As regras xerais (RX-x) de emprego dos repositorios son:

RX-1. Poderanse facer todos os *commits* que se consideren oportunos en ambos repositorios, coa única restrición de aportar un comentario suficientemente descritivo do *commit* en cuestión.

RX-2. Non será preciso, salvo casos excepcionais, facer uso de ramas (*branches*) que non sexan a principal (*master*), posto que tan só traballará cos repositorios un único individuo.

No caso do código fonte, deben cumprirse tamén as seguintes regras específicas (RE-x):

RE-1. O remate dun *sprint* de traballo asociarase cunha nova *tag* no repositorio. Este etiquetado non ten por que coincidir coa versión do produto, pero será o máis recomendable.

RE-2. Non se impoñen restricións respecto a cando definir as distintas versións do produto. O desenvolvedor terá a potestade de decidir cando etiquetar o software cunha nova versión en base ao valor engadido que poida aportar a devandita ao público.

RE-3. Para nomear unha nova versión da aplicación deberase seguir o patrón `v<decimal>` (por exemplo: `v1.0`, `v4.1`, etc). Cada nova versión deberá etiquetarse cunha cifra superior a calquera das versións anteriores, aínda que non se require ningún tipo de linearidade.

Para os demais ECSs (de deseño e de xestión), posto que non se necesita acceder a versións anteriores, emprégase un sistema distribuído máis sinxelo, que consiste nun directorio **Dropbox** (`https://www.dropbox.com/`). Créanse subdirectorios segundo se estima oportuno para garantir a boa accesibilidade dos elementos da configuración. Aínda que en principio non se necesita recuperar versións anteriores, o servizo Dropbox integra un control de versións propio que permite acceder ao estado anterior dos elementos con criterio de data e hora.

Liñas base e seguemento dos cambios

Faise uso de liñas base para o control do código fonte e da memoria. As liñas base coinciden con cada unha das versións do repositorio. No caso do código fonte, tamén existe unha liña base de saída de cada un dos *sprints*. A liña base de referencia en cada momento é sempre a máis recente. Cada unha das liñas base debe ser accesible de forma individual en calquera momento do ciclo de vida do proxecto.

Diferénciase entre control de cambios maiores e cambios menores. Enténdese por **cambios maiores** aqueles que implican asignación de tarefas. Os **cambios menores** son todos aqueles que non implican asignación de traballo aos membros do equipo. Calquera cambio que se queira realizar sobre unha liña base debe seguir o procedemento exposto a continuación.

Para cambios menores, é suficiente con realizar un novo *commit* no que quede constancia do cambio introducido. A propia información do rexistro de Git (identificador único, data, hora e usuario) xunto cun comentario descritivo é dabondo neste tipo de situacións.

Cando se trate de cambios maiores, debe determinarse que recursos empregar e engadirse unha nova entrada á **táboa de xestión de cambios** do directorio Dropbox. Dita táboa reflicte a seguinte información de cada cambio:

- `ID`: identificador único da solicitude de cambio maior.
- `Data de solicitude`: data de solicitude do cambio.
- `Solicitante`: persoas que solicitan o cambio.
- `Motivo`: razón pola que se debe realizar o cambio.
- `Descrición`: explicación das tarefas a realizar no cambio.
- `Estimación do custo`: aproximación dos recursos necesarios para a realización do cambio.
- `Estado de aceptación`: pode tomar dous valores: `aceptado` ou `rechazado`.
- `Responsable`: se se asigna o cambio, deberá determinarse un responsable.
- `Encargado(s) de realización`: persoas ás que lle son asignadas tarefas para a realización do cambio.
- `Data límite`: data límite para ter realizado o cambio.
- `Estado de aprobación`: pode tomar os seguintes valores: `aprobado` ou `denegado`.
- `Motivo da aprobación/denegación`: argumentos da resolución a favor ou en contra.
- `Data de resolución da aprobación/denegación`: data na que se formaliza a aprobación o denegación do traballo realizado.
- `tag versión`: rematada a implementación do cambio, *tag* que identifica a súa liña base.

3.2. Plan de xestión de riscos

A presente sección detalla o plan de xestión dos posibles riscos do proxecto, que inclúe os procesos relacionados coa planificación, a identificación e a análise de riscos, as respostas aos mesmos, e o seu seguimento e control ao longo do proxecto. A maioría destes procesos foron actualizándose de xeito paralelo á evolución do proxecto.

Un risco do proxecto é un evento ou condición incerta que, de producirse, ten un efecto positivo ou negativo sobre, canto menos, un obxectivo do mesmo. Un risco pode ter unha ou máis causas e, se se produce, un ou máis impactos. Se ocorre algún destes eventos incertos, pode haber un impacto sobre o cronograma, o custo, o alcance ou a calidade do proxecto. Os obxectivos deste plan son aumentar a probabilidade e o impacto dos eventos positivos, e diminuír a probabilidade e o impacto dos eventos adversos.

3.2.1. Metodoloxía

En CMMI ou PMBOK[1] a xestión de riscos é un área de proceso con entidade propia e á que xeralmente se lle presta moita atención. En Scrum, sen embargo, a aproximación á xestión de riscos é menos explícita, pero isto non quere dicir que non se lle preste atención, simplemente o enfoque é diferente.

O problema dos enfoques tradicionais é que non son áxiles. Esta xestión é pesada e burocrática e rara vez se leva coa disciplina necesaria e coa actualización adecuada para que sexa efectiva. Ademais, o habitual é centrarse naqueles riscos graves e evidentes, que son moi perigosos pero tamén son comúns a todos os proxectos e poucas veces se manifestan.

En Scrum substitúese a xestión de riscos explícita pola **xestión de riscos continua**. Unha das preguntas que se deben responder de xeito diario ao longo do proxecto é: que impedimentos atopaches? Responder a esta pregunta día a día é, sen dúbida, unha maneira implícita de xestionar os riscos do proxecto. Outro excelente momento para detectar riscos é durante o **Sprint Retrospective**[2], que permite simplificar todos os riscos relacionados coa comunicación co cliente e os requisitos.

[1] CMMI e PMBOK son guías amplamente coñecidas que recollen unha serie de fundamentos e métodos tradicionais para a xestión e dirección de proxectos.

[2] O Sprint Retrospective é unha reunión efectuada polo equipo de desenvolvemento ao remate de cada *sprint* co obxectivo de mellorar de maneira continua a súa produtividade e a calidade do produto que está desenvolvendo, analizando como foi a súa maneira de traballar durante a iteración, por que está conseguindo ou non os obxectivos e por que o incremento de produto que acaba de demostrar ao cliente era o que el esperaba ou non.

Neste proxecto, adaptando do mellor xeito posible a filosofía Scrum, realizouse unha xestión de riscos continua, que se foi documentando nunha folla de cálculo, `identificacionRiscos.ods` (ver Apéndice B). Os procesos levados a cabo nesta xestión foron os seguintes:

1. Identificación de activos, ameazas e riscos.
2. Análise cuantitativa dos riscos.
3. Aceptación ou rexeitamento dentro do plan de xestión de riscos para cada un dos riscos identificados.
4. Determinación da estratexias de acción para cada risco aceptado.
5. Seguimento e control de cada risco.

Estratexia de identificación de riscos

Para identificar os riscos empregouse o **método clásico e sistemático**: coméza-se identificando os activos do proxecto; a continuación, extráense cales son as ameazas sobre ditos activos e, finalmente, identifícanse os riscos derivados de ditas ameazas. Esta técnica aplicouse de forma continua ao longo do proxecto, de xeito individual, nas reunións semanais co Product Owner e durante os Sprint Retrospective.

Outras técnicas de identificación, como a tormenta de ideas (*brainstorming*), tiveron que ser descartadas posto que están ideadas para equipos de desenvolvemento de máis dunha persoa.

Estratexia de análise cuantitativa de riscos

Para poder avaliar o nivel de exposición aos riscos, recórrese ao emprego dunha **matriz de probabilidade e impacto** propia, que se amosa na Táboa 3.1. De xeito previo ao análise de dita matriz, compre definir algúns conceptos:

- **Probabilidade**: a expectativa de ocorrencia real dun evento.
- **Impacto**: o efecto que a ocorrencia dun evento tería no desenvolvemento do proxecto, en termos de custo, esforzo e duración total do mesmo.
- **Nivel de exposición ao risco**: é o produto Impacto × Probabilidade, de maneira que será este parámetro (agregado dos anteriores) o que goberne a xestión dos riscos do proxecto.

A matriz clasifica os riscos segundo a súa probabilidade de ocorrencia e segundo o impacto sobre a consecución dos obxectivos aos que afectan, é dicir, en que medida

Nivel de exposición ao risco				
		Probabilidade		
		Alta	Media	Baixa
Impacto	Alto	Alto	Alto	Medio
	Medio	Alto	Medio	Baixo
	Baixo	Medio	Baixo	Baixo

Táboa 3.1: Matriz de probabilidade e impacto cos valores do nivel de exposición ao risco.

se deixan de lograr ditos obxectivos en caso de producirse o risco. Na matriz emprégase unha escala de intervalos propia para definir tanto a probabilidade (baixa, media, alta) como o impacto (baixo, medio, alto). Os límites que definen ditos intervalos son:

- Probabilidade:
  Baixa: <= 10 % (pouco probable)
  Media: entre o 10 % e o 40 % (probable)
  Alta: >= 40 % (moi probable)
- Impacto:
  Baixo: <= 20 %
  Medio: entre o 20 % e o 50 %
  Alto: >= 50 %

Estratexia de planificación da resposta aos riscos

Considérase que un risco é aceptado dentro deste plan de xestión de riscos no momento no que se realiza a planificación da resposta ao mesmo. Para cada risco selecciónanse unha ou varias das seguintes tácticas:

- **Evitar**: cambiar as condicións orixinais dese evento para eliminar totalmente o risco identificado.
- **Mitigar**: levar a cabo accións concretas que diminúan, ou ben a probabilidade de aparición da ameaza, ou ben o seu impacto se acaba materializándose. Á súa vez diferenciamos dúas estratexias derivadas:
  - **Previr**: cambiar o plan de xestión do proxecto para reducir a probabilidade de que se desencadee o evento non desexado.
  - **Minimizar** ou conter: preparar un plan de acción que só se executa baixo determinadas condicións ou sinais de advertencia, co fin de reducir o impacto do risco.

- **Transferir**: trasladar o impacto negativo dunha ameaza, xunto coa propiedade da resposta, a un terceiro. Non se elimina o risco, senón que a responsabilidade do mesmo é asumida por outro. Esta estratexia soe implicar unha contrapartida económica.

- **Aceptar**: non cambiar o plan de xestión orixinal para facer fronte a un risco, ben porque non se considera prioritario ou non se atopa ningunha outra estratexia de resposta adecuada. Esta é a estratexia seleccionada para a maioría de riscos cun nivel de exposición baixo.

3.2.2. Activos e ameazas

A Táboa 3.2 contén os activos identificados ata a data, mentres que a Táboa 3.3 contén as ameazas relacionadas con ditos activos. Esta información pode ser consultada en detalle no arquivo `identificacionRiscos.ods`.

Identificador	Descripción	Clasificación
ACT-1	Desenvolvedor	Persoas
ACT-2	Clientes	Persoas
ACT-3	Product Owner	Persoas
ACT-4	Beta testers	Persoas
ACT-5	Equipo de traballo	Hardware
ACT-6	Dispositivo móbil de desenvolvemento	Hardware
ACT-7	Sistema Operativo	Software
ACT-8	Software de desenvolvemento	Software
ACT-9	Software de xestión	Software
ACT-10	Código fonte da app	Software
ACT-11	Pila de Produto	Información
ACT-12	Memoria do proxecto	Información
ACT-13	Documento de riscos	Información
ACT-14	Servidor Gitlab	Servizos
ACT-15	Servizos de comunicación	Servizos
ACT-16	Rede da USC	Redes
ACT-17	Rede eléctrica	Instalacións
ACT-18	Oficina de traballo	Instalacións
ACT-19	Mobiliario	Instalacións

Táboa 3.2: Activos do plan de xestión de riscos.

Identificador	Activos (ACT-x)	Descrición	Clasificación
AMZ-1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19	Falta de dispoñibilidade	Accidentais por persoas
AMZ-2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19	Desastre artificial	Artificiais
AMZ-3	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19	Desastre natural	Naturais
AMZ-4	1, 2, 3, 10, 11, 12	Indecisión	Accidentais por persoas
AMZ-5	1, 2, 3, 4, 10, 12	Insatisfacción	Deliberadas por persoas
AMZ-6	1, 2, 3, 4, 10, 11, 12	Falta de comunicación	Accidentais por persoas
AMZ-7	1, 10, 11, 12	Falta de experiencia coas tecnoloxías	Accidentais por persoas
AMZ-8	1, 2, 3, 4, 5, 6	Deterioro do hardware	Artificiais
AMZ-9	1, 2, 10, 11, 12	Mala planificación	Accidentais por persoas
AMZ-10	1, 2, 3, 4	Conflitos interpersoais	Deliberadas por persoas
AMZ-11	1, 2, 3, 4, 10, 12	Desmotivación	Accidentais por persoas
AMZ-12	1, 2, 3, 4, 10	Escaseza de coñecementos legais	Accidentais por persoas
AMZ-13	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	Velocidade dos avances tecnolóxicos	Deliberadas por persoas

Táboa 3.3: Ameazas do plan de xestión de riscos.

3.2.3. Especificación de riscos

Preséntase a continuación o catálogo final de riscos do proxecto. Buscando compactar a información achégase unha táboa por cada risco identificado, que inclúe unha descrición do mesmo, as súas dependencias e a información relativa á súa análise cuantitativa, planificación da resposta e seguimento e control. No campo *Indicadores* defínense os indicadores e cuantificadores de disparo para comezar a por en práctica as medidas de acción definidas no campo superior. Omítese información como a probabilidade e o impacto para definir directamente a exposición. Compre lembrar que a información non incluída nestas táboas pode consultarse no documento adxunto `identificacionRiscos.ods`.

Índice de riscos

- [RSC-1]: Cambios nos requisitos.
- [RSC-2]: Atrasos na planificación temporal.
- [RSC-3]: Perda de información.
- [RSC-4]: Mala implementación ou deseño inadecuado.
- [RSC-5]: Manifestación de erros.
- [RSC-6]: Atoamento por ausencia de persoal clave.
- [RSC-7]: Expectativas do cliente non acadadas.
- [RSC-8]: Incumprimento da lei.
- [RSC-9]: Mala xestión de riscos.
- [RSC-10]: Equipo inutilizado.
- [RSC-11]: Entrega de versión errónea.

Detalle dos riscos

RSC-1 **Cambios nos requisitos**	
Versión:	1.3
Autor:	Fernando Estévez
Descrición:	Efectúanse cambios nos requisitos, ben porque o cliente cambia de parecer respecto dalgunha funcionalidade, ou ben por mala captura dos mesmos.
Activos:	ACT-1, ACT-2, ACT-3, ACT-10, ACT-11
Ameazas:	AMZ-4, AMZ-5, AMZ-6
Exposición:	Alta
Estratexia:	Mitigar (previr e minimizar)
Acción:	Por un lado, como medida preventiva séguese a filosofía Scrum tanto no que respecta á xestión dos requisitos como á interacción co cliente. Deste xeito, os requisitos vanse especificando ao longo do proxecto na Pila de Produto e o cliente pode seguir de preto a súa evolución. Ademais, como medida de continxencia, definiuse un bo plan de xestión da configuración que, xunto coa tolerancia a cambios ofrecida por Scrum, axudarán á boa xestión deste tipo de eventos.
Indicadores:	Aplicadas medidas preventivas desde o inicio do proxecto. As medidas de continxencia serán tomadas en canto se produza unha solicitude de cambio.
Incidencias:	Solicitudes de cambio do 11/04, do 19/05 e do 02/06 de 2017.

RSC-2 **Atrasos na planificación temporal**	
Versión:	1.0
Autor:	Fernando Estévez
Descrición:	O proxecto non segue a planificación prevista por motivos diversos: lentitude nos procesos de documentación, descoñecemento das tecnoloxías, lentitude na fase de probas, etc.
Activos:	ACT-1, ACT-2, ACT-3, ACT-10, ACT-12
Ameazas:	AMZ-1 – AMZ-11
Exposición:	Alta
Estratexia:	Mitigar (previr)
Acción:	De novo Scrum aporta un bo sistema preventivo. Coa metodoloxía áxil aplicada, o equipo é o último responsable de aceptar os prazos e de comprometerse coa cantidade de características a implementar durante o *sprint*. Ninguén pode impor prazos que no sexan realistas pois o equipo ten a potestade para non aceptalos. Respecto da burocracia, recordar que Scrum aposta pola documentación lixeira.
Indicadores:	Aplicadas medidas desde o inicio do proxecto.
Incidencias:	Sen incidencias.

RSC-3 **Perda de información**	
Versión:	1.0
Autor:	Fernando Estévez
Descrición:	Pérdese información relativa ao proxecto, ben sexa código fonte ou documentación, por motivos diversos (ver ameazas).
Activos:	ACT-1 – ACT-3, ACT-10 – ACT-14
Ameazas:	AMZ-2, AMZ-3, AMZ-7, AMZ-8
Exposición:	Alta
Estratexia:	Evitar
Acción:	Integrouse un plan de xestión da configuración, sinxelo pero consistente, que evita este risco ao manter toda a información do proxecto de forma local no equipo de traballo e replicada na nube (ver Sección 3.1).
Indicadores:	Aplicadas medidas desde o inicio do proxecto.
Incidencias:	Sen incidencias.

RSC-4 **Mala implementación ou deseño inadecuado**	
Versión:	1.0
Autor:	Fernando Estévez
Descrición:	Prodúcense problemas na codificación do software ou ben durante o deseño, que poden carrexar a insatisfacción do cliente ou o incumprimento dalgún requisito.
Activos:	ACT-1 – ACT-3, ACT-10

RSC-4 **Mala implementación ou deseño inadecuado** *(cont.)*	
Ameazas:	AMZ-6, AMZ-7
Exposición:	Alta
Estratexia:	Mitigar (previr e minimizar)
Acción:	Definiuse un plan de formación que tratará de evitar estes problemas no relativo á falta de experiencia coas tecnoloxías. O Sprint Review proposto en Scrum e a demostración que se realiza durante o mesmo, alertan rapidamente da carencia dun deseño adecuado. Durante a Sprint Retrospective tamén se poden detectar partes da aplicación que deben ser refactorizadas. Scrum é unha metodoloxía preparada para o cambio e a refactorización.
Indicadores:	Aplicadas medidas desde o inicio do proxecto.
Incidencias:	Sen incidencias.

RSC-5 **Manifestación de erros**	
Versión:	1.0
Autor:	Fernando Estévez
Descrición:	Detección de erros na aplicación, ben sexa aínda na fase de desenvolvemento ou xa durante a distribución e explotación. Cobran especial interese os erros por problemas de compatibilidade.
Activos:	ACT-1 – ACT-3, ACT-10
Ameazas:	AMZ-7, AMZ-11, AMZ-13
Exposición:	Alta
Estratexia:	Mitigar (previr)
Acción:	Deseñouse un plan de probas completo e esixente, que inclúe, como particularidade, a posta en marcha dunha beta privada que permita identificar por enriba de todo erros de compatibilidade, os cales non poden ser probados cun plan de probas tradicional. A maiores, o plan de formación paliará a introdución de outros erros lóxicos por parte do equipo de desenvolvemento.
Indicadores:	Aplicadas medidas desde o inicio do proxecto.
Incidencias:	Sen incidencias.

RSC-6 **Atoamento por ausencia de persoal clave**	
Versión:	1.0
Autor:	Fernando Estévez
Descrición:	Imposibilidade de cumprimento da planificación por dependencias con *stakeholders* non dispoñibles cando son necesitados.
Activos:	ACT-1 – ACT-4, ACT-10, ACT-12
Ameazas:	AMZ-1, AMZ-6, AMZ-10, AMZ-11
Exposición:	Alta
Estratexia:	Evitar

RSC-6 **Atoamento por ausencia de persoal clave** *(cont.)*	
Acción:	Scrum segue o Manifesto Áxil e, por tanto, o principio de antepor a colaboración co cliente sobre a negociación de contratos. Para elo Scrum pon en todo proxecto un representante dos intereses do cliente, o Product Owner, e ademais permite e persegue a colaboración e a comunicación con todos os involucrados no proxecto durante os Sprint Reviews. Neste proxecto en concreto adicouse tempo a elaborar un bo plan de comunicacións.
Indicadores:	Aplicadas medidas desde o inicio do proxecto.
Incidencias:	Sen incidencias.

RSC-7 **Expectativas do cliente non acadadas**	
Versión:	1.0
Autor:	Fernando Estévez
Descrición:	O produto non cumpre coas expectativas do cliente.
Activos:	ACT-1 – ACT-3, ACT-10
Ameazas:	AMZ-1, AMZ-6, AMZ-10, AMZ-11
Exposición:	Media
Estratexia:	Evitar
Acción:	Tómanse as medidas relativas á interacción cos clientes expostas nos riscos RSC-1, RSC-4 e RSC-6.
Indicadores:	Aplicadas medidas desde o inicio do proxecto.
Incidencias:	Sen incidencias.

RSC-8 **Incumprimento da lei**	
Versión:	1.0
Autor:	Fernando Estévez
Descrición:	O produto non cumpre coa normativa legal vixente de protección de datos ou propiedade intelectual.
Activos:	ACT-1, ACT-10
Ameazas:	AMZ-12
Exposición:	Media
Estratexia:	Evitar
Acción:	En base aos coñecementos básicos legais do equipo de desenvolvemento, rexeitarase calquera requisito, funcionalidade, tarefa ou actividade que incite a mínima dúbida respecto da súa legalidade.
Indicadores:	Aplicadas medidas desde o inicio do proxecto.
Incidencias:	Sen incidencias.

RSC-9 **Mala xestión de riscos**	
Versión:	1.0
Autor:	Fernando Estévez

RSC-9 **Mala xestión de riscos** *(cont.)*	
Descrición:	Durante o proxecto realízase unha identificación ou xestión dos riscos ineficiente, que pode implicar a manifestación doutros riscos non contemplados.
Activos:	ACT-1, ACT-10
Ameazas:	AMZ-7, AMZ-11
Exposición:	Media
Estratexia:	Aceptar
Acción:	Asúmese o risco. Parece moi improbable que nun proxecto desta envergadura se manifesten outros riscos non contemplados neste plan e, de manifestárense, a metodoloxía de traballo de seguro facilitará o seu control.
Indicadores:	Non aplica.
Incidencias:	Sen incidencias.

RSC-10 **Equipo inutilizado**	
Versión:	1.0
Autor:	Fernando Estévez
Descrición:	Un equipo de traballo (computador ou dispositivo móbil) queda inutilizado.
Activos:	ACT-1, ACT-5 – ACT-9
Ameazas:	AMZ-2, AMZ-3, AMZ-8, AMZ-13
Exposición:	Baixa
Estratexia:	Mitigar (previr e minimizar)
Acción:	Se se estraga o dispositivo móbil de probas empregarase outro de uso particular. De estragarse o equipo de traballo, o CiTIUS proporcionará un equipo de escritorio que o substituirá. Documentarase o proceso de posta a punto do entorno de traballo (instalación de sistema operativo, software de desenvolvemento e xestión, bibliotecas e outras dependencias, etc) co fin de axilizar o proceso de volta á normalidade.
Indicadores:	Non aplica.
Incidencias:	Sen incidencias.

RSC-11 **Entrega dunha versión errónea**	
Versión:	1.0
Autor:	Fernando Estévez
Descrición:	Entrégase ao cliente unha versión incorrecta ou inconsistente do produto.
Activos:	ACT-1 – ACT-3, ACT-10
Ameazas:	AMZ-6, AMZ-11
Exposición:	Baixa
Estratexia:	Mitigar (previr) e aceptar.

RSC-11 **Entrega dunha versión errónea** *(cont.)*	
Acción:	Tómanse as medidas relativas ao control de versións expostas nos riscos RSC-1 e RSC-3. De non ser dabondo o control de versións debido a un erro humano, o impacto será insignificante en base á confianza e proximidade do cliente.
Indicadores:	Aplicadas medidas preventivas desde o inicio do proxecto. As medidas de continxencia serán tomadas en canto se avaríe o dispositivo en cuestión.
Incidencias:	Sen incidencias.

3.3. Planificación temporal

Nesta sección recóllese a planificación temporal do proxecto. Preséntase, en primeiro lugar a Estrutura de Descomposición do mesmo, seguida polo cronograma completo e actualizado.

3.3.1. Estrutura de Descomposición do Traballo

A Figura 3.1 amosa unha representación da Estrutura de Descomposición do Traballo (EDT) do proxecto. A descrición de cada un dos cadros da EDT pódese consultar no dicionario disposto a continuación.

Dicionario da EDT

1. **Análise**. Bloque adicado á etapa de análise, da cal se destacan estas catro subtarefas:

   1.1. **Definición de obxectivos e alcance**. Fase inicial de estudo do problema, determinando de forma clara e unívoca os obxectivos que se perseguen co proxecto e cuxa consecución marcará a finalización exitosa do mesmo.

   1.2. **Especificación de requisitos**. Posto que se decidiu optar por unha metodoloxía áxil baseada en Scrum, a especificación de requisitos é continua (ver Capítulo 2).

   1.3. **Selección da metodoloxía**. Fase na que se decide a metodoloxía de traballo a seguir. Isto implica non só elixir o modelo, Scrum, senón tamén adaptalo a este proxecto en concreto.

   1.4. **Selección das tecnoloxías**. Fase na que se determina que tecnoloxías, métodos e algoritmia empregar para afrontar os distintos retos.

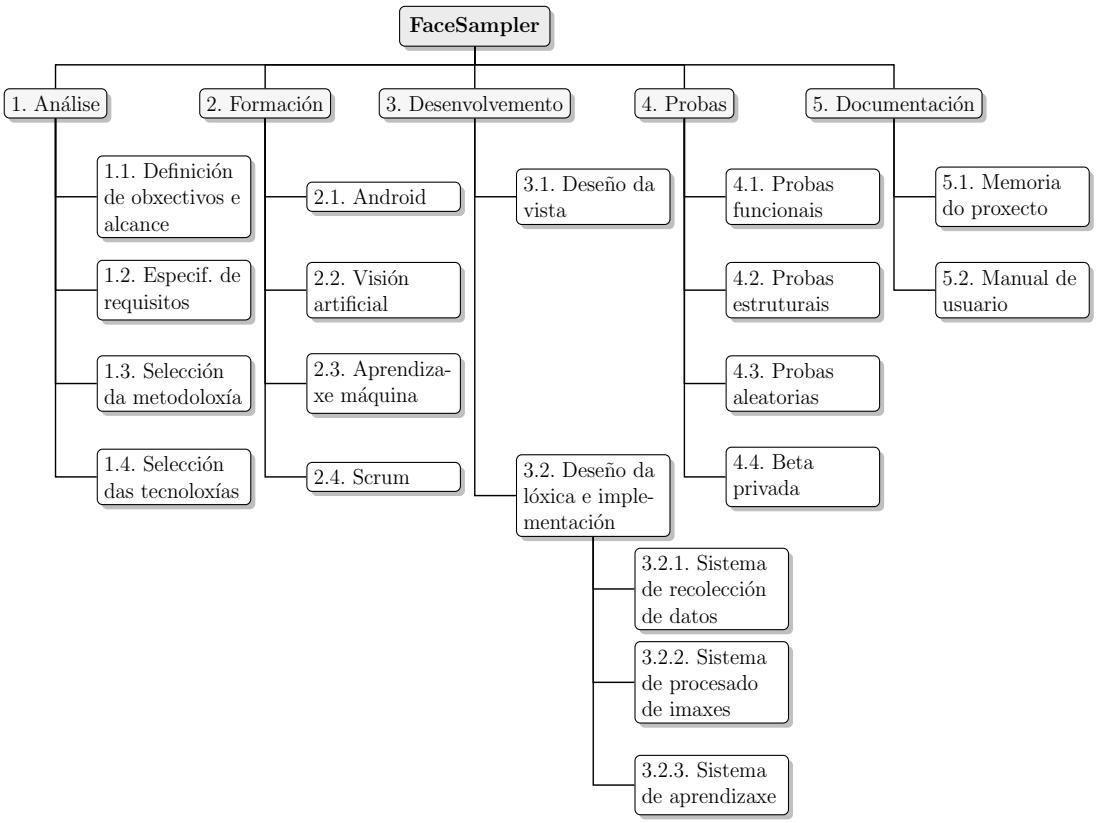


Figura 3.1: EDT do proxecto.

2. **Formación**. Bloque adicado á aprendizaxe das tecnoloxías empregadas. Cada un dos bloques derivados deste na EDT correspóndese con unha tecnoloxía que se necesita aprender. Recoméndase a lectura da Sección 3.4.1, onde se describe cada unha das mesmas.

3. **Desenvolvemento**. Este bloque comprende o deseño e a codificación da aplicación. Ao empregarse unha metodoloxía áxil, ambas etapas están moi solapadas.

   3.1. **Deseño da vista**. Deseño inicial das vistas da aplicación, así como da navegación entre as mesmas.

   3.2. **Deseño da lóxica e implementación**. Diferéncianse tres grandes subsistemas dentro da aplicación. Recoméndase a lectura do Capítulo 5, onde se describe cada un en detalle.

4. **Probas**. Bloque adicado á aplicación do plan de probas. Cada un dos bloques derivados deste na EDT correspóndese cunha compoñente da estratexia de dito plan. Recoméndase a lectura da Sección 3.6, onde se describe dita estratexia en detalle.

5. **Documentación**. Bloque correspondente á elaboración da seguinte documentación:

   5.1. **Memoria do proxecto**. Memoria técnica do proxecto, elaborada paralelamente ao desenvolvemento do mesmo, na que se recollen os detalles de tódalas etapas abordadas, así como as conclusións finais.

   5.2. **Manual de usuario**. Documento onde se detalla o modo de emprego da aplicación.

3.3.2. Cronograma

A Figura 3.2 amosa o cronograma final do proxecto representado mediante un diagrama de Gantt. As frechas indican as dependencias de realización entre tarefas ou grupos de tarefas. As tarefas coloreadas en vermello conforman o camiño crítico do mesmo. Coñecer o camiño crítico permite saber cales son as actividades que deben realizarse para que nada se atrase e se cumpran os prazos estimados.

O tempo medio de dedicación ao proxecto foi de aproximadamente **14 horas á semana**. Pese a todo, ao longo dos meses produciuse unha gran variación nesta media en función de factores alleos ao proxecto, maioritariamente académicos. Existiron porén etapas de dedicación practicamente nula, como o período de exames de decembro e xaneiro, e outras de maior entrega, como os meses de febreiro, maio e xuño, onde se adicaron ata 25 horas semanais.

Todos os *sprints* se planificaron para equivaleren a **40 horas** de traballo, aproximadamente. Isto traduciuse en *sprints* de entre 2 e 3 semanas, en función da dispoñibilidade dos recursos e da prioridade do desenvolvemento en cada instante.

En total, adicáronse a este proxecto **419 horas**. Recoméndase a consulta do documento adxunto `planificacionTemporal.ods` para ver os detalles da repartición das horas. A continuación apórtase unha breve descrición de cada unha das actividades máis relevantes incluídas no diagrama de Gantt.

- **Análise inicial**. Primeira etapa á que se lle adicou unha semana de traballo, 19 horas, e que coincide co bloque 1 da EDT anterior. Cabe destacar que, aínda que a especificación de requisitos sexa constante, realizouse a tarefa *Especificación inicial de requisitos* para identificar aqueles requirimentos máis importantes e nos que se debía comezar a traballar de inmediato.
- ***Sprint* 1**. Coincide co inicio do desenvolvemento da aplicación Android. O tempo total adicado ascendeu a 42 horas, repartidas ao longo de 3 semanas. Implementouse un detector facial en primeiro plano e en tempo real partindo do aprendido con OpenCV e Java (ver Apéndice D) durante o

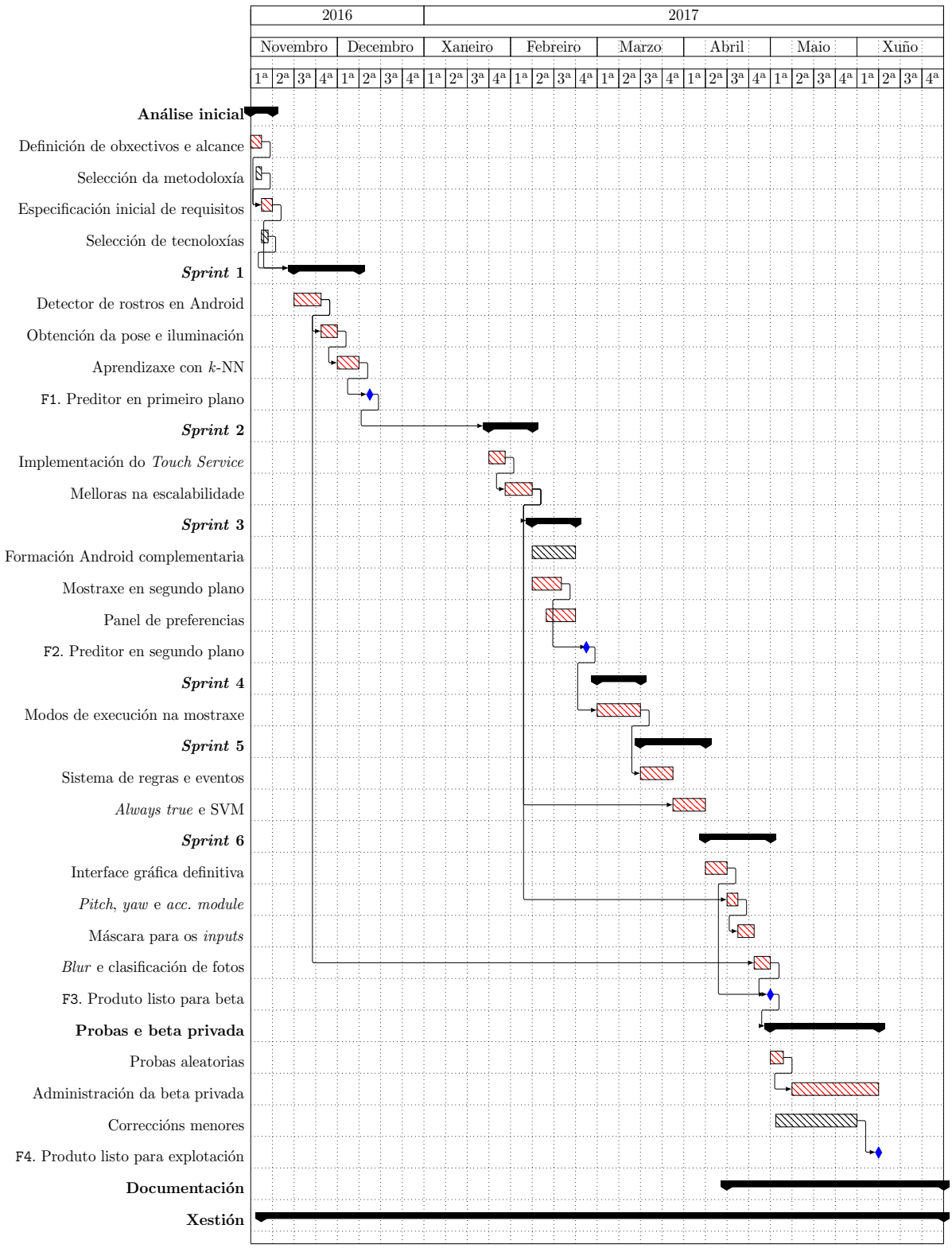


Figura 3.2: Diagrama de Gantt coa planificación do proxecto.

anterior mes de xullo. Tamén se comezaron a obter datos do entorno que servirían como entradas da aprendizaxe. Concretamente, obtívose a pose do dispositivo e a iluminación do entorno. Finalmente, introduciuse o primeiro sistema de aprendizaxe supervisado, baseado no algoritmo (k-NN). Recoméndase ler a Sección 4.1 para coñecer máis detalles.

- ***Sprint* 2**. A súa duración foi de 2 semanas, 36 horas de traballo. Neste *sprint* realizáronse melloras na arquitectura da aplicación para favorecer a súa escalabilidade respecto da captura de información e as entradas do sistema de aprendizaxe. Tamén se comezou a obter información relacionada coa manipulación do dispositivo, concretamente as pulsacións de pantalla (*Touch Service*).

- ***Sprint* 3**. Durou un total de 40 horas, que se traducen en 2 semanas de traballo. Integrouse un sistema de mostraxe capaz de executarse en segundo plano mentres o usuario desenvolve outras tarefas. Tamén se creou un panel de preferencias básicas relativas á aprendizaxe. Foi necesaria formación complementaria en Android para levar a cabo ambas tarefas.

- ***Sprint* 4**. Adicáronselle 2 semanas de traballo, un total de 36 horas. Implementouse o sistema de modos de execución da mostraxe que se explica na Sección 4.2.2.

- ***Sprint* 5**. A súa duración foi unha vez máis de 36 horas, traducidas en 3 semanas de traballo. Púxose en funcionamento a permutación entre os modos de execución definidos no *sprint* anterior, mediante o emprego dun sistema de regras e eventos disparadores. Tamén se revisou o deseño do módulo de aprendizaxe para que fose escalable a novos algoritmos, e integráronse *Always true* e SVM (ver Sección 4.1).

- ***Sprint* 6**. Durou 3 semanas, traballando un total de 40,5 horas. Fixéronse melloras significativas: implementouse a interface de usuario definitiva, modificouse o conxunto de datos de entrada para empregar as rotacións *pitch* e *yaw* e o módulo da aceleración, implementouse unha máscara para poder deshabilitar estes *inputs* e comezou a detectarse o desenfoque (*blur*) nas imaxes e a facerse unha clasificación das mesmas na galería do dispositivo.

- **Probas e beta privada**. O esforzo adicado a esta etapa foi de 29,5 horas, aínda que se prolongou durante 5 semanas. En primeiro lugar realizáronse un conxunto de probas aleatorias. Posteriormente, iniciouse unha beta privada da aplicación, que se prolongou por un mes. Paralelamente, fóronse detectando fallos e corrixindo aqueles erros que requirían de pouco tempo para solucionalos. Recoméndase ler a estratexia do plan de probas da Sección 3.6 para comprender a finalidade destas tarefas.

- **Documentación**. Esta actividade correspóndese directamente co bloque 5

da EDT anterior. Comezouse coa documentación do proxecto a principios do mes de maio e adicáronselle un total de 102 horas, repartidas ao longo de 8 semanas.

- **Xestión**. A xestión do proxecto levouse a cabo de forma continua. En total, estímanse 38 horas de esforzo, destinadas a tarefas como identificación e xestión de riscos, xestión de requisitos mediante a Pila de Produto, control de cambios, planificación temporal, etc.

Fitos

Scrum representa moi claramente o significado de fito na xestión de proxectos. Divide o proxecto nun conxunto de bloques ou partes que deben ser executadas dentro dun prazo e custo estimados. A Táboa 3.4 describe cada un dos fitos do proxecto. Estes marcan datas, eventos e logros importantes dentro do desenvolvemento do proxecto e axudan como guía para avaliar o rendemento do mesmo.

Data	ID	Descrición
12/15/2016	`F1`	Aplicación Android executándose completamente en primeiro plano con aprendizaxe supervisado e predición do momento oportuno para a toma de mostras.
24/02/2017	`F2`	A aprendizaxe é configurable e a mostraxe pode executarse en segundo plano durante tempo prolongado (días, semanas...).
17/02/2017	`F3`	A interface de usuario é intuitiva e amigable. A mostraxe realízase de forma eficiente, optimizando recursos, e as mostras almacénanse de xeito local na memoria do dispositivo.
21/02/2017	`F4`	A aplicación xa foi probada nun grupo reducido de usuarios, con variedade de dispositivos e versións Android e está lista para pasar a unha distribución masiva.

Táboa 3.4: Fitos do proxecto.

3.4. Plan de xestión de Recursos Humanos

Scrum propón tres roles diferenciados que deben formalizarse: Product Owner, Scrum Master e Scrum Team. O **Product Owner** é o representante principal do cliente, e a única persoa autorizada para decidir sobre as funcionalidades e características do produto. O **Scrum Master** pode considerarse como o líder do equipo de desenvolvemento (**Scrum Team**), e encárgase de fomentar e instruír sobre os principios áxiles de Scrum.

Neste caso, por tratarse dun equipo de traballo cun único desenvolvedor, os roles de Scrum Master e Scrum Team combínanse nunha única persoa, de xeito que esta asume todas as responsabilidades de ambos, adquirindo a potestade para:

- Decidir sobre como aplicar os principios propostos por este marco de traballo (Scrum Master).
- Asegurar o cumprimento da planificación e a realización das tarefas propostas (Xefe de Proxecto).
- Identificar e analizar os requirimentos do proxecto (Analista).
- Realizar o deseño da arquitectura e dos compoñentes do sistema (Deseñador).
- Codificar o software (Programador).
- Probar o produto (Encargado de probas).

De novo por tratarse dun equipo unipersoal, non será necesario incluír neste plan ningunha medida de **xestión de conflitos ou incidencias** de carácter interpersoal.

3.4.1. Plan de formación

As necesidades tecnolóxicas deste proxecto requiriron da elaboración dun plan formativo que contemplase todas aquelas tecnoloxías novidosas para o equipo. Coméntanse a continuación as medidas aplicadas para favorecer a aprendizaxe das mesmas, explicando exclusivamente a estratexia formativa. Para coñecer en detalle cada unha destas tecnoloxías, recoméndase a lectura da Sección 4.3, onde se achega unha descrición técnica de cada unha delas e a xustificación da súa elección no proxecto.

Android

Aínda que xa se fixera previamente unha aplicación sinxela para esta plataforma, o nivel de coñecemento era máis ben baixo. Saber traballar con Android non só implica programar co seu SDK, senón tamén dominar o IDE Android Studio, as vías de publicación e as tecnoloxías de *testing*. O aprendizaxe de Android baseouse nas seguintes actividades:

- Realización do curso *Study Jams: Android Fundamentals*, ofertado polo GDG Spain. Esta actividade realizouse de xeito previo ao comezo do proxecto e tivo unha duración de 50 horas, 10 de elas presenciais.

- Seguimento de tutoriais e experimentación autónoma coas principais tecnoloxías de probas: JUnit, Mockito, Expresso, Firebase, etc. Realizouse ao longo da fase desenvolvemento do proxecto, baixo necesidade.
- Lectura das guías oficiais de Google e experimentación autónoma coa Google Play Console para a publicación e distribución da aplicación, así como o seu mantemento. Realizouse de xeito paralelo á propia publicación.

Visión artificial

Neste proxecto implementouse un sistema de detección facial que emprega as funcionalidades ofrecidas pola biblioteca OpenCV, que se basea no método de Viola-Jones [7]. A aprendizaxe neste caso levouse a cabo no mes de xullo de 2016, de xeito previo ao comezo do proxecto, durante o desfrute dunha bolsa de verán ofertada polo CiTIUS. Implementouse un prototipo do sistema de detección en Java, como unha aplicación de escritorio. No Apéndice D fálase sobre este prototipo.

Aprendizaxe máquina e sensorización

No avance do proxecto, foi necesario estudar os métodos de aprendizaxe supervisado e clasificación que se ían empregar: k-NN e SVM (ler Sección 4.1). Para as dúbidas, a estratexia foi pedir axuda a Roberto Iglesias. Por exemplo, cando se desexou integrar o emprego do cuaternión para coñecer a pose do dispositivo, organizouse unha titoría de 3 horas con Roberto.

Desenvolvemento áxil e Scrum

Como achegamento inicial ás metodoloxías áxiles e Scrum, na primeira semana do proxecto realizouse o curso *online Gestión de proyectos con metodologías ágiles y enfoques Lean*, de aproximadamente 8 horas de traballo, ofertado por Miríada X (Fundación Telefónica). Para as dúbidas puntuais, a estratexia consistiu en realizar consultadas no libro "Scrum and XP from the trenches" [10] ou na web.

3.5. Plan de xestión de custos

O seguinte plan de xestión de custos inclúe os procesos involucrados na estimación, preparación do presuposto e control de custos de forma que o proxecto se

poida completar dentro do presuposto aprobado. Debido á natureza académica do mesmo, os custos manexados neste son puramente teóricos.

A estimación de custos permite ao equipo de desenvolvemento obter unha aproximación dos custos dos recursos necesarios para completar cada actividade do cronograma. A preparación do presuposto implica sumar os custos estimados das actividades do cronograma ou paquetes de traballo individuais para establecer unha liña base de custo total, a fin de medir o rendemento do proxecto. Por último, o control de custos busca as causas das variacións positivas e negativas producidas durante o seu desenvolvemento, e forma parte do control de cambios.

3.5.1. Estimación de custos

Nesta sección exponse a estimación do presuposto do proxecto, explicando previamente como se realizou, indicando as ferramentas, criterios e información de respaldo necesarios.

Ferramentas e criterios

Distínguese entre custos **directos** e **indirectos**. Os primeiros correspóndense coas inversións en recursos necesarios no proxecto, que poden ser humanos ou materiais. Para todo o material adquirido establécese unha **vida útil**, de xeito que nos custos finais do proxecto non se reflicte o prezo total de cada adquisición, senón a cantidade proporcional ao período que abarca o mesmo. Os segundos, os custos indirectos, son aqueles que non se poden atribuír de forma directa ao proxecto, como o gastos en electricidade, auga, acceso a Internet e mantemento do espazo de traballo.

Todas as cantidades monetarias aproxímanse cun nivel de exactitude de dous decimais na contabilidade dos custos. As unidades de medida utilizadas son:

- Para os recursos humanos:
  Horas/persoa: cuantificación do traballo.
  Euros/hora: cuantificación do custo.
- Para os recursos materiais:
  Unidades: cantidade.
  Euros/unidade: custo.

Información de respaldo da estimación

A estimación foi desenvolta partindo da planificación temporal do proxecto (ver Sección 3.3). A maiores, empregáronse os seguintes documentos de apoio:

- Guía salarial do sector TIC en Galicia, 2015-2016 [14].
- Acta do Consello de Goberno da USC, sesión do 11/11/2008 [15].
- Resolución do 13 de febreiro de 2017, da Dirección General de Empleo (BOE-A-2017-1868) [16].

Resumo da estimación

Como se explicaba no plan de xestión de RRHH da Sección 3.4, o equipo de desenvolvemento está formado por un único individuo. Posto que este realiza variedade de tarefas sempre relativas a procesos do ámbito da enxeñaría de software, asígnaselle o rol de **Enxeñeiro de Software I+D+i *junior***. Segundo a guía salarial comentada anteriormente [14], estímase o salario medio anual (S_{EA}) deste rol na provincia de A Coruña en 18000 € brutos. Tendo en conta que a xornada de traballo efectivo anual (T_{EA}) para o ano 2017 é de 1787 horas [16], podemos calcular o custo por hora do enxeñeiro (C_E/h):

$$C_E/\text{h} = \frac{S_{EA}}{T_{EA}} = \frac{18000\ €}{1787\ \text{h}} = 10,07\ €/\text{h}$$

Coñecendo o número de horas totais destinadas á elaboración deste proxecto (T_P), 419 horas, extraídas da planificación temporal da Sección 3.3, podemos obter o custo total do enxeñeiro (C_E):

$$C_E = C_E/\text{h} \times T_P = 10,07\ €/\text{h} \times 419\ \text{h} = \mathbf{4219{,}33}\ €$$

Como material de traballo, empregáronse os seguintes dispositivos, adquiridos no seu día a estes prezos:

- PC *Ultrabook* **Lenovo Yoga 3 Pro** ..... 1059,95 €
- *Smartphone* **Bq Aquaris E5s** ..... 164,95 €

Suponse un tempo de vida útil (T_{VU}) de **4 anos** para o PC, e de **3 anos** para o *smartphone*. O custo de cada dispositivo no proxecto (C_d) virá dado pola seguinte fórmula, onde C_{Td} é o custo total do mesmo:

$$C_d = \frac{C_{Td} \times T_P}{T_{VU}}$$

Deste xeito, o custo do PC (C_{PC}) é:

$$C_{PC} = \frac{1059{,}95\ € \times 419\ \text{h}}{4\ \text{anos} \times 8760\ \text{h/ano}} = \mathbf{12{,}67\ €}$$

E para o dispositivo móbil (C_{sphone}):

$$C_{sphone} = \frac{164{,}95\ € \times 419\ \text{h}}{3\ \text{anos} \times 8760\ \text{h/ano}} = \mathbf{2{,}63\ €}$$

Todas as ferramentas software empregadas son libres, polo que non existen custos asociados ás mesmas. O mesmo pasa co sistema operativo. Recoméndase a lectura da Sección 4.3, onde se fala en detalle dos dispositivos e ferramentas utilizados.

Por último, tal e como certifica a Secretaria Xeral da Universidade de Santiago de Compostela [15], aplicarase o **20 % dos custos directos** para calcular os custos indirectos daquelas actividades realizadas por investigadores pertencentes a departamentos ou institucións da Universidade, entre os que se inclúen o CiTIUS e o Departamento de Electrónica e Computación.

En ausencia de máis factores a ter en conta, preséntase a continuación a estimación definitiva dos custos do proxecto:

Custos directos	4234,63 €
Enxeñeiro de Software I+D+i	4219,33 €
PC *Ultrabook* Lenovo Yoga 3 Pro	12,67 €
Smartphone Bq Aquaris E5s	2,63 €
Custos indirectos	846,93 €
+20 % dos custos directos	846,93 €
TOTAL	**5081,56 €**

3.5.2. Plan de control de custos

Para asegurar o éxito do proxecto, é necesario establecer como se realizará o seguimento dos custo. A maior parte do presuposto destínase á paga de salarios, polo que o seguimento fará énfases na terminación das tarefas nos prazos previstos para evitar sobrecustos.

É de esperar que se xeren solicitudes de cambio nos custos, que serán xestionadas polo procedemento xeral de solicitude de cambios do proxecto, definido na

Sección 3.1. Dado que na planificación temporal se chega a un nivel de detalle semanal, estes cambios deberán ser documentados, canto menos, sempre que impliquen unha variación de máis de 3 xornadas laborais (24 horas).

3.6. Plan de probas

O obxectivo deste plan é presentar información sobre a calidade do produto, atopar defectos e facilitar información para a toma de decisións. Preténdese que, a medio prazo, estas accións permitan evitar a manifestación de erros e aumentar a confianza no nivel de calidade. O plan foise adaptando conforme avanzaba o proxecto. Preséntase agora a versión final do mesmo.

3.6.1. Estratexia de construción

Desde que se comezou a traballar en FaceSampler sabíase que sería inviable aplicar un plan de probas exhaustivo. As restricións temporais foron sempre consideradas como o principal impedimento para acadar este fin. Sen embargo, posteriormente detectouse outro obstáculo do mesmo calibre: a falta de experiencia coas ferramentas de *testing* do ecosistema Android. Por estes dous condicionantes, o plan de probas foi evolucionando de xeito que se sacrificou cobertura en profundidade (métodos e clases) para lograr acadala en amplitude (módulos e todo o sistema integrado).

A filosofía aplicada foi a de procurar a axilidade, garantindo sempre uns mínimos niveis de cobertura, que serán comentados nesta mesma sección. Aplicando tamén as pautas de Scrum, as probas fóronse deseñando e executando ao longo do proxecto, o cal permitiu obter unha realimentación periódica que axudou a regular o plan.

Estableceuse unha cobertura mínima do 50 % para cada unha das funcionalidades resultado de cada *sprint* do proxecto. De non acadarse dita cobertura, o seguinte *sprint* é replanificado para seguir traballando naquelas funcionalidades que non logren dito mínimo. Nunca se continuará coa implementación de novas características se non se cumpre este criterio. Deste xeito, sexa cal sexa o estado do produto, vai ter unha cobertura en amplitude de polo menos un 50 % na última das súas versións (ver Sección 3.1).

A maiores, compre destacar o fortemente ligadas que están as funcionalidades deste proxecto. Isto implica que, en moitas ocasións, ao probar unha nova característica se estean probando de xeito indirecto as anteriores, o cal axudará a incrementar a cobertura das mesmas. Isto unido ao feito de que en Scrum se abarcan primeiro as funcionalidades máis priorizadas polo cliente axuda a que aquelas

máis importantes estean antes e mellor probadas, acadando graos de cobertura superiores aos daquelas outras menos prioritarias.

Non obstante, as dependencias entre funcionalidades tamén poden ser desfavorables a este plan. Por exemplo, probar unha nova funcionalidade que depende de outra anterior, por moi probada que esta estea, sempre pode carrexar problemas. Compre tomar decisións durante as fases de proba de cada *sprint* respecto de que compoñentes *mockear* (empregar obxectos simulados) para garantir o correcto *testeo* da nova funcionalidade introducida.

Acadando a cobertura desexada

Considerouse (e confirmouse posteriormente) que unha cobertura do 50 % soe ser facilmente acadable realizando **probas funcionais**, ou de caixa negra. Seguindo a filosofía áxil, tomáronse como probas funcionais aquelas establecidas como probas de aceptación para cada un dos casos de uso do sistema. Evidentemente, efectuáronse estas probas por duplicado, a segunda vez en presencia do cliente. En todas elas aplícanse as técnicas de **conxectura de erros**[1] e **clases de equivalencia**[2] para xerar suficientes casos de proba. FaceSampler caracterízase por ter a maior parte das funcionalidades "ligadas" á interface gráfica, de xeito que as súas accións repercuten na pantalla do dispositivo móbil. Grazas a isto, moitas destas probas puideron realizarse monitorando a execución normal da aplicación e forzando a casuística oportuna para obter os resultados desexados por pantalla.

Este plan conta tamén con **probas estruturais**, ou de caixa branca, pensadas para asegurar o correcto funcionamento de obxectos complexos, con moitas dependencias ou que non poden ser facilmente cubertos coas probas funcionais descritas no parágrafo anterior.

Abordando o problema da compatibilidade

Un problema que estivo moi presente ao longo do proxecto foi o da compatibilidade da aplicación con distintos dispositivos e versións de Android. Existe un requisito non funcional `[RNF-14]` no que se definen os niveis mínimos aceptables de compatibilidade (ver Sección 2.3). O problema, tal e como se recolle no plan de riscos (Sección 3.2) co risco "Manifestación de erros" `[RSC-5]`, é que resulta moi difícil calcular o grao de compatibilidade acadado pola aplicación sen poder

[1] Técnica de probas consistente na elaboración previa dunha lista de erros non contemplados anteriormente que serve para xerar novos casos de proba.

[2] Técnica de probas que consiste na división do campo de entrada para obter estados válidos e non válidos do sistema.

probala de xeito exhaustivo en diversas versións e modelos do mercado. De feito, é posible que a aplicación pase unha proba, funcional ou estrutural, no dispositivo móbil de desenvolvemento pero non en outros. Como solución, integrouse no plan un conxunto de **probas aleatorias** e a posta en explotación dunha versión **beta privada**.

As probas aleatorias realizáronse ao remate da etapa de implementación. Para isto, empregouse a ferramenta web **Firebase Test Lab** de Google [17]. Esta, dispón dun laboratorio de dispositivos de probas, tanto físicos como virtuais. Deste xeito, permite seleccionar con que dispositivos, niveis da API de Android, orientacións de pantalla e configuracións rexionais se desexa probar a aplicación. Logo, por cada configuración seleccionada execútase unha proba automática que analiza a estrutura da interface gráfica e simula actividades de usuarios para explorala de maneira autónoma.

De xeito posterior a esa execución de probas aleatorias, planificouse a posta en marcha dunha versión beta da aplicación limitando o acceso a un grupo de persoas de confianza. A idea consistiu en que os voluntarios empregasen FaceSampler durante un período de tempo suficiente como para lograr obter 220 mostras do seu rostro. Paralelamente, o equipo de desenvolvemento seguiu o comportamento da aplicación en cada dispositivo a través da **Google Play Console** [18], detectando as incidencias (fallos e ANRs) que puidesen ocorrer.

O obxectivo de incluír estas medidas no plan de probas é dobre. Por un lado, axudarán a obter un informe de compatibilidade que permita verificar o requisito non funcional asociado `[RNF-14]`. Ademais, tamén servirán como reforzo ás probas funcionais e estruturais definidas anteriormente, co cal se espera mellorar a cobertura das diversas funcionalidades da aplicación.

3.6.2. Riscos do plan de probas

O único risco identificado como propio deste plan é a incapacidade para lograr as coberturas mínimas definidas, o cal podería desencadear os riscos `RSC-2`, `RSC-4` ou `RSC-5` do proxecto. Este risco, polo tanto, xa está integrado de xeito indirecto nos plans de acción destes outros tres riscos globais.

A estratexia específica pensada para este risco é preventiva. Cando se detecte un atraso na planificación `[RSC-2]` por lentitude nas probas, replanificarase o seguinte *sprint* para continuar traballando sobre os requisitos actuais, tal e como se comentaba na sección anterior. Scrum engade facilidade a esta estratexia, posto que o equipo é o último responsable de aceptar os prazos e de comprometerse coa cantidade de características a implementar durante o *sprint* e o cliente pode seguir a evolución de preto do produto, sendo consciente das adversidades manifestadas.

Para maior detalle, recoméndase a lectura da Sección 3.2, de planificación de riscos.

3.6.3. Definición de probas

Exponse a continuación a especificación das probas deseñadas para a aplicación durante os sucesivos *sprints* do desenvolvemento.

Probas funcionais

Como se explicaba previamente, as probas funcionais a realizar ao longo do proxecto xa están definidas como probas de aceptación dos casos de uso. Poden ser consultadas na Sección 2.4.

Probas estruturais

Como probas estruturais, defínense as seguintes:

`PE-1` **Proba de conexión de servizos Android**	
Autor:	Fernando Estévez
Propósito:	Verificar a correcta conexión cos distintos *services* de Android integrados na aplicación.
Funcións abordadas:	Inicio dos distintos servizos por medio de solicitudes de tipo *start* ou *bindings*.
Técnicas aplicadas:	Conxectura de erros e clases de equivalencia
Casos de proba:	`CP-1`, `CP-2`
Criterio(s) de paso:	Compróbase que os distintos servizos son iniciados e poden ser empregados con normalidade.
Criterio(s) de fallo:	Algún dos servizos non é iniciado correctamente, lanzando a excepción `java.util.concurrent.TimeoutException`.
Requisitos do entorno:	Esta proba require de dependencias Android, concretamente do `Context` da aplicación, polo que deben definirse casos de proba instrumentais [19].

`PE-2` **Proba dos algoritmo de aprendizaxe**	
Autor:	Fernando Estévez
Propósito:	Verificar que os algoritmos de aprendizaxe k-NN e SVM realizan predicións atinadas en casos libres de dúbida.
Funcións abordadas:	Toma de decisións con k-NN e SVM.
Técnicas aplicadas:	Conxectura de erros e clases de equivalencia

PE-2 **Proba dos algoritmo de aprendizaxe** *(cont.)*	
Casos de proba:	CP-3, CP-4, CP-5, CP-6, CP-7, CP-8, CP-9, CP-10
Criterio(s) de paso:	Realizar todas as predicións de xeito previsible.
Criterio(s) de fallo:	Fallar unha ou máis predicións.
Requisitos do entorno:	Requírese de dependencias Android, concretamente do `Context` da aplicación, polo que deben definirse casos de proba instrumentais [19].

PE-3 **Proba do xestor de modos de execución**	
Autor:	Fernando Estévez
Propósito:	Comprobar que as permutacións entre modos de execución da mostraxe (Normal e *Low Energy*) se realizan de forma exitosa en resposta aos distintos eventos.
Funcións abordadas:	Cambio entre modos desde o `BroadcastReceiver` do que dispón o `ExecutionModeHandler`.
Técnicas aplicadas:	Conxectura de erros e clases de equivalencia
Casos de proba:	CP-11, CP-12, CP-13, CP-14, CP-15, CP-16, CP-17, CP-18, CP-19
Criterio(s) de paso:	Permutar entre os modos Normal e *Low Energy* tal e como se espera.
Criterio(s) de fallo:	Non permutar baixo o previsto nalgún dos casos de proba.
Requisitos do entorno:	Esta proba require de dependencias Android, concretamente do `Context` da aplicación, polo que deben definirse casos de proba instrumentais [19].

Probas aleatorias

Pola súa natureza impredicible, non foi necesario realizar un deseño previo das accións a levar a cabo neste tipo de probas. Non obstante, cumpriu facer unha selección das distintas combinacións de dispositivos e versións Android a empregar nas mesmas. A estratexia aplicada consistiu en procurar a maior diversidade posible, tendo presente que se pretendía estimar a compatibilidade da aplicación. A Táboa 3.5 recolle tódalas configuracións empregadas.

No Capítulo 6 pódese consultar como foi a execución, así como os detalles das incidencias reportadas derivadas da mesma.

3.6.4. Definición de casos de proba

Expóñense a continuación os casos de proba xerados a partir das probas estruturais definidas na sección previa.

Proba	Dispositivo	API Android	Versión Android	Orientación
PA-1	Samsung Galaxy Note 3	19	4.4	Retrato
PA-2	Google Nexus 7	21	5.0	Retrato
PA-3	Google Nexus 9	21	5.0	Paisaxe
PA-4	Sony Xperia Z3	21	5.0	Retrato
PA-5	OnePlus One	22	5.1	Retrato
PA-6	Google Nexus 6	22	5.1	Retrato
PA-7	Samsung Galaxy Note 4	22	5.1	Retrato
PA-8	Motorola Moto G4	23	6.0	Retrato
PA-9	Samsung Galaxy S7 edge	23	6.0	Retrato
PA-10	Google Nexus 5	23	6.0	Retrato
PA-11	Google Nexus 5	23	6.0	Paisaxe
PA-12	Sony Xperia X	23	6.0	Retrato
PA-13	Samsung Galaxy S7	24	7.0	Retrato
PA-14	Google Pixel	25	7.1	Retrato
PA-15	Google Nexus 5X	25	7.1	Retrato
PA-16	Google Nexus 5X	25	7.1	Paisaxe

Táboa 3.5: Plan de probas aleatorias.

PE-1 Proba de conexión de servizos

CP-1 testStartService	
Responsable:	Fernando Estévez
Datos precargados:	Non hai datos precargados.
Método a invocar:	`Context.startService(i: Intent): ComponentName`
Entradas:	`ForegroundSamplingService.class`
Saídas:	Comprobación de que o servizo en cuestión, `ForegroundSamplingService`, está en execución mediante o uso dun `ActivityManager`.
Dependencias:	• `android.content.Context` • `android.support.test.rule.ServiceTestRule` • `android.app.ActivityManager`

CP-2 testWithBoundService	
Responsable:	Fernando Estévez
Datos precargados:	Non hai datos precargados.
Método a invocar:	`ServiceTestRule.bindService(i: Intent): IBinder`
Entradas:	`CameraService.class`
Saídas:	`CameraServie.isReady() == true`
Dependencias:	• `android.content.Context` • `android.support.test.rule.ServiceTestRule` • `android.os.IBinder`

PE-2 Proba dos algoritmos de aprendizaxe

CP-3 testKNNHasFaceFalseSimple	
Responsable:	Fernando Estévez

Datos precargados:	• State s1 {
	pattern = {0, 0, 0, 0, 0}
	label = 1 }
	• State s2 {
	pattern = {1, 1, 1, 1, 1}
	label = 0 }

Método a invocar:	Knn.getInstance().hasFace(s: State, model: ArrayList<State>, context: Context): boolean
Entradas:	• State s { pattern = {1, 1, 1, 1, 1} } • ArrayList<State> model = {s1, s1, s1, s1, s2, s2}
Saídas:	false
Dependencias:	• android.content.Context • model.learning.algorithms.knn.Knn

CP-4 testKNNHasFaceTrueSimple	
Responsable:	Fernando Estévez

Datos precargados:	• State s1 {
	pattern = {0, 0, 0, 0, 0}
	label = 1 }
	• State s2 {
	pattern = {1, 1, 1, 1, 1}
	label = 0 }

Método a invocar:	Knn.getInstance().hasFace(s: State, model: ArrayList<State>, context: Context): boolean
Entradas:	• State s { pattern = {0, 0, 0, 0, 0} } • ArrayList<State> model = {s1, s1, s1, s1, s2, s2}
Saídas:	true
Dependencias:	• android.content.Context • model.learning.algorithms.knn.Knn

CP-5 testKNNHasFaceFalse	
Responsable:	Fernando Estévez

CP-5 testKNNHasFaceFalse *(cont.)*	

Datos precargados:	• State s1 {
	pattern = {0, 0, 0, 0, 0}
	label = 1 }
	• State s2 {
	pattern = {0, 0.1, 0.2, 0.05, 0}
	label = 1 }
	• State s3 {
	pattern = {0, 0.02, 0, 0.25, 0}
	label = 0 }
	• State s4 {
	pattern = {0.8, 1, 0.9, 1, 1}
	label = 0 }
	• State s5 {
	pattern = {1, 1, 1, 1, 1}
	label = 0 }

Método a invocar:	Knn.getInstance().hasFace(s: State, model: ArrayList<State>, context: Context): boolean
Entradas:	• State s {pattern = {0.1, 0, 0.1, 0.05, 0}} • ArrayList<State> model = {s1, s2, s3, s4, s5}
Saídas:	false
Dependencias:	• android.content.Context • model.learning.algorithms.knn.Knn

CP-6 testKNNHasFaceTrue	
Responsable:	Fernando Estévez

Datos precargados:	• State s1 {
	pattern = {0, 0, 0, 0, 0}
	label = 1 }
	• State s2 {
	pattern = {0, 0.1, 0.2, 0.05, 0}
	label = 1 }
	• State s3 {
	pattern = {0, 0.02, 0, 0.25, 0}
	label = 1 }
	• State s4 {
	pattern = {0.8, 1, 0.9, 1, 1}
	label = 0 }
	• State s5 {
	pattern = {1, 1, 1, 1, 1}
	label = 0 }

CP-6 testKNNHasFaceTrue *(cont.)*	
Método a invocar:	`Knn.getInstance().hasFace(s: State, model: ArrayList<State>, context: Context): boolean`
Entradas:	• `State s { pattern = {0.1, 0, 0.1, 0.05, 0} }` • `ArrayList<State> model = {s1, s2, s3, s4, s5}`
Saídas:	`true`
Dependencias:	• `android.content.Context` • `model.learning.algorithms.knn.Knn`

CP-7 testSVMHasFaceFalseSimple	
Responsable:	Fernando Estévez
Datos precargados:	• `State s1 { pattern = {0, 0, 0, 0, 0} label = 1 }` • `State s2 { pattern = {1, 1, 1, 1, 1} label = 0 }`
Método a invocar:	`OpenCvSVM.getInstance().hasFace(s: State, model: ArrayList<State>, context: Context): boolean`
Entradas:	• `State s { pattern = {1, 1, 1, 1, 1} }` • `ArrayList<State> model = {s1, s1, s1, s1, s2, s2}`
Saídas:	`false`
Dependencias:	• `android.content.Context` • `model.learning.algorithms.svm.OpenCvSVM`

CP-8 testSVMHasFaceTrueSimple	
Responsable:	Fernando Estévez
Datos precargados:	• `State s1 { pattern = {0, 0, 0, 0, 0} label = 1 }` • `State s2 { pattern = {1, 1, 1, 1, 1} label = 0 }`
Método a invocar:	`OpenCvSVM.getInstance().hasFace(s: State, model: ArrayList<State>, context: Context): boolean`

CP-8 testSVMHasFaceTrueSimple *(cont.)*	
Entradas:	• State s { pattern = {0, 0, 0, 0, 0} } • ArrayList<State> model = {s1, s1, s1, s1, s2, s2}
Saídas:	true
Dependencias:	• android.content.Context • model.learning.algorithms.svm.OpenCvSVM

CP-9 testSVMHasFaceFalse	
Responsable:	Fernando Estévez

Datos precargados:	• State s1 {
	pattern = {0, 0, 0, 0, 0}
	label = 1 }
	• State s2 {
	pattern = {0, 0.1, 0.2, 0.05, 0}
	label = 1 }
	• State s3 {
	pattern = {0, 0.02, 0, 0.25, 0}
	label = 0 }
	• State s4 {
	pattern = {0.8, 1, 0.9, 1, 1}
	label = 0 }
	• State s5 {
	pattern = {1, 1, 1, 1, 1}
	label = 0 }

Método a invocar:	OpenCvSVM.getInstance().hasFace(s: State, model: ArrayList<State>, context: Context): boolean
Entradas:	• State s { pattern = {0.1, 0, 0.1, 0.05, 0} } • ArrayList<State> model = {s1, s2, s3, s4, s5}
Saídas:	false
Dependencias:	• android.content.Context • model.learning.algorithms.svm.OpenCvSVM

CP-10 testSVMHasFaceTrue	
Responsable:	Fernando Estévez

CP-10 testSVMHasFaceTrue *(cont.)*	

Datos precargados:	• State s1 {
	pattern = {0, 0, 0, 0, 0}
	label = 1 }
	• State s2 {
	pattern = {0, 0.1, 0.2, 0.05, 0}
	label = 1 }
	• State s3 {
	pattern = {0, 0.02, 0, 0.25, 0}
	label = 1 }
	• State s4 {
	pattern = {0.8, 1, 0.9, 1, 1}
	label = 0 }
	• State s5 {
	pattern = {1, 1, 1, 1, 1}
	label = 0 }

Método a invocar:	OpenCvSVM.getInstance().hasFace(s: State, model: ArrayList<State>, context: Context): boolean
Entradas:	• State s { pattern = {0.1, 0, 0.1, 0.05, 0} } • ArrayList<State> model = {s1, s2, s3, s4, s5}
Saídas:	true
Dependencias:	• android.content.Context • model.learning.algorithms.svm.OpenCvSVM

PE-3 Proba do xestor de modos de execución

CP-11 testNothingHappens	
Responsable:	Fernando Estévez
Datos precargados:	Mostraxe executándose en modo normal (Execution.MODE_NORMAL).
Método a invocar:	ExecutionModeHandler.registerBroadcastReceiver(): BroadcastReceiver
Entradas:	Non hai entradas.
Saídas:	Consulta a ExecutionModeHandler.getCurrentMode() que devolve Execution.MODE_NORMAL.
Dependencias:	• android.content.Context • android.content.Intent

CP-12 testUserWantsLEMode	
Responsable:	Fernando Estévez
Datos precargados:	Mostraxe executándose en modo normal (`Execution.MODE_NORMAL`).
Método a invocar:	`ExecutionModeHandler.registerBroadcastReceiver(): BroadcastReceiver`
Entradas:	Envíase un `Intent` simulando unha solicitude do usuario para pasar a modo *low energy*.
Saídas:	Consulta a `ExecutionModeHandler.getCurrentMode()` que devolve `Execution.MODE_LOW_ENERGY`.
Dependencias:	• `android.content.Context` • `android.content.Intent`

CP-13 testUserWantsNormalMode	
Responsable:	Fernando Estévez
Datos precargados:	Mostraxe executándose en modo normal (`Execution.MODE_NORMAL`).
Método a invocar:	`ExecutionModeHandler.registerBroadcastReceiver(): BroadcastReceiver`
Entradas:	Envíase un `Intent` simulando unha solicitude do usuario para pasar a modo normal.
Saídas:	Consulta a `ExecutionModeHandler.getCurrentMode()` que devolve `Execution.MODE_NORMAL`.
Dependencias:	• `android.content.Context` • `android.content.Intent`

CP-14 testUserWantsLEModeAndNormalModeAgain	
Responsable:	Fernando Estévez
Datos precargados:	Mostraxe executándose en modo normal (`Execution.MODE_NORMAL`).
Método a invocar:	`ExecutionModeHandler.registerBroadcastReceiver(): BroadcastReceiver`
Entradas:	Envíase un `Intent` simulando unha solicitude do usuario para pasar a modo *low energy* e posteriormente outro para tornar a modo normal.
Saídas:	Consulta a `ExecutionModeHandler.getCurrentMode()` que devolve `Execution.MODE_NORMAL`.
Dependencias:	• `android.content.Context` • `android.content.Intent`

CP-15 testUserWantsNormaModeAndLEModeThen	
Responsable:	Fernando Estévez

CP-15 testUserWantsNormaModeAndLEModeThen *(cont.)*	
Datos precargados:	Mostraxe executándose en modo normal (`Execution.MODE_NORMAL`).
Método a invocar:	`ExecutionModeHandler.registerBroadcastReceiver(): BroadcastReceiver`
Entradas:	Envíase un `Intent` simulando unha solicitude do usuario para pasar a modo normal (pese a que ese sexa o modo vixente) e posteriormente outro para pasar a modo *low energy*.
Saídas:	Consulta a `ExecutionModeHandler.getCurrentMode()` que devolve `Execution.MODE_LOW_ENERGY`.
Dependencias:	• `android.content.Context` • `android.content.Intent`

CP-16 testUserPresent	
Responsable:	Fernando Estévez
Datos precargados:	Mostraxe executándose en modo normal (`Execution.MODE_NORMAL`).
Método a invocar:	`ExecutionModeHandler.registerBroadcastReceiver(): BroadcastReceiver`
Entradas:	Envíase un `Intent` coa acción `ACTION_USER_PRESENT`, simulando o desbloqueo de pantalla por parte do usuario.
Saídas:	Consulta a `ExecutionModeHandler.getCurrentMode()` que devolve `Execution.MODE_NORMAL`.
Dependencias:	• `android.content.Context` • `android.content.Intent`

CP-17 testScreenOff	
Responsable:	Fernando Estévez
Datos precargados:	Mostraxe executándose en modo normal (`Execution.MODE_NORMAL`).
Método a invocar:	`ExecutionModeHandler.registerBroadcastReceiver(): BroadcastReceiver`
Entradas:	Envíase un `Intent` coa acción `ACTION_SCREEN_OFF`, simulando o bloqueo de pantalla por parte do usuario.
Saídas:	Consulta a `ExecutionModeHandler.getCurrentMode()` que devolve `Execution.MODE_LOW_ENERGY`.
Dependencias:	• `android.content.Context` • `android.content.Intent`

CP-18 testStopFromNormal	
Responsable:	Fernando Estévez

CP-18 testStopFromNormal *(cont.)*	
Datos precargados:	Mostraxe executándose en modo normal (`Execution.MODE_NORMAL`).
Método a invocar:	`ExecutionModeHandler.registerBroadcastReceiver(): BroadcastReceiver`
Entradas:	Envíase un `Intent` coa acción `Execution.ACTION_STOP`, simulando a orde de detección por parte do usuario.
Saídas:	Consulta a `ExecutionModeHandler.getCurrentMode()` que devolve `Execution.MODE_NORMAL`.
Dependencias:	• `android.content.Context` • `android.content.Intent`

CP-19 testStopFromLE	
Responsable:	Fernando Estévez
Datos precargados:	Mostraxe executándose en modo normal (`Execution.MODE_LOW_ENERGY`).
Método a invocar:	`ExecutionModeHandler.registerBroadcastReceiver(): BroadcastReceiver`
Entradas:	Envíase un `Intent` coa acción `Execution.ACTION_STOP`, simulando a orde de detección por parte do usuario.
Saídas:	Consulta a `ExecutionModeHandler.getCurrentMode()` que devolve `Execution.MODE_LOW_ENERGY`.
Dependencias:	• `android.content.Context` • `android.content.Intent`

3.7. Plan de xestión das comunicacións

Nesta sección descríbese brevemente a forma na que se planificaron, estruturaron, monitoraron e controlaron as comunicacións do proxecto.

3.7.1. Xestión dos interesados

Cando falamos de interesados (*stakeholders*) referímonos a todas aquelas persoas que participan nalgún momento no proxectou ou cuxos intereses se poden ver afectados polas actividades levadas a cabo no mesmo. Recordemos que no Capítulo 1 xa se comentaron todas as partes participantes no proxecto, resumíndoas na Táboa 1.4.

Ademais dos *stakeholders* habituais, neste proxecto levouse a cabo unha beta privada, tal e como comentabamos no plan de probas da sección previa. Isto implicou levar un control de cada un dos voluntarios que decidiron participar na mesma.

Para elo, mantívose actualizado un documento en formato `ODS` (folla de cálculo), onde se gardou a información de cada un deles: nome, correo electrónico, teléfono e dirección ou despacho (opcional). A maiores, tamén se gardou información dos seus dispositivos móbiles, de cara a documentar as probas (modelo e versión de Android). Por motivos de privacidade e protección de datos non se proporciona maior información respecto dos interesados.

3.7.2. Reunións

Seguindo a filosofía Scrum, ao longo do proxecto leváronse a cabo reunións de catro tipos:

- **Planificación do *sprint***. Antes de iniciar o seguinte *sprint* efectuouse sempre unha reunión previa entre o equipo e o Product Owner, representando ao cliente. A saída desta reunión é a planificación do *sprint*, que abarca a lista de obxectivos e tarefas, a orde na que se abordarán e as datas de inicio e remate. Scrum tamén recomenda que se defina un responsable por cada tarefa, sen embargo, debemos lembrar que neste proxecto o equipo de traballo estivo composto por un só individuo.
- **Reunión diaria (*daily meeting*)**. Scrum propón efectuar unha reunión dun máximo de 15 minutos cada día entre os membros do equipo co fin de facer unha sincronización entre eles, falando do que fixeron o día anterior, os problemas que tiveron e o que pensan facer no día actual. Neste caso, polas restricións de persoal, esta reunión fíxose, pero de xeito simulado. O desenvolvedor adicou os 5 primeiros minutos de cada xornada de traballo a revisar o feito no día anterior e planificar a xornada vixente.
- **Demo (*sprint* review)**. Ao remate de cada *sprint* efectuouse sempre unha reunión na que o equipo de traballo lle ensinaba os resultados do mesmo ao cliente. Aquí era onde se decidía se se podía avanzar ou non se estaban acadando as expectativas do cliente.
- **Retrospectiva (*sprint retrospective*)**. Este tipo de *meeting* tamén se realizou ao remate de cada *sprint*. O equipo de traballo facía un repaso do que fora o *sprint* e procuraba identificar e avaliar problemas concretos. De novo, ao igual que na reunión diaria, tan só participaba o desenvolvedor. Compre lembrar a importancia desta reunión na identificación e xestión dos riscos do proxecto, tal e como se comentou na Sección 3.2.

Capítulo 4

Proposta

Este capítulo pretende recoller aquelas decisións tomadas en base aos obxectivos e requisitos do proxecto que dan solución a aspectos técnicos moi específicos do problema. Convén que sexan presentadas de xeito previo ao deseño do sistema, posto que este último verase condicionado en base a moitas destas decisións.

En primeiro lugar, na Sección 4.1, presentarase unha proposta teórica que resolve o problema da toma de decisións respecto a cando capturar unha mostra do usuario. Posteriormente, na Sección 4.2, exporase a estratexia seguida para facer unha xestión eficiente dos recursos do dispositivo. Por último, na Sección 4.3 comentaranse as tecnoloxías seleccionadas para o desenvolvemento.

4.1. A toma de decisións

O terceiro dos obxectivos do proxecto `[OBX-3]` consistía en lograr dotar á aplicación coa capacidade de aprender de cara á toma de decisións na captura de mostras do usuario. A maiores, o segundo obxectivo `[OBX-2]` matizaba que a captura non podía requirir da participación activa do mesmo.

Para acadar estas metas, preséntase a proposta da Figura 4.1. O móbil debe aprender que "estados" son axeitados para tomar unha imaxe coa súa cámara frontal. No noso caso, o **estado** do dispositivo vén representado por un **patrón** coa seguinte información de contexto:

- **Rotación *pitch***. Mide a rotación en torno ao eixe x do móbil, representado con cor verde na Figura 4.2. É unha compoñente vital, posto que nos axuda a predicir se a cámara está a encadrar verticalmente ao usuario.
- **Rotación *yaw***. Mide a rotación en torno ao eixe z do dispositivo (de cor azul na Figura 4.2). Axúdanos a saber se a cámara está inclinada con

respecto ao eixe y, o que podería orixinar que non se detecten rostros por non aparecer coa mesma orientación que o dispositivo.

- **Iluminación do entorno**. A iluminación é outro factor importante a ter en conta cando tomamos a foto. Se a escena carece de luz, posiblemente a imaxe sexa demasiado escura como para detectar nela ningún rostro.
- **Inversa do tempo transcorrido desde o último toque de pantalla**. Se o usuario interactúa coa pantalla, é probable que se estean dando as condicións ideais para obter unha mostra.
- **Módulo da aceleración**. Permite diferenciar cando o móbil está en repouso e cando suceden movementos bruscos, os cales introducirán desenfoque na imaxe.

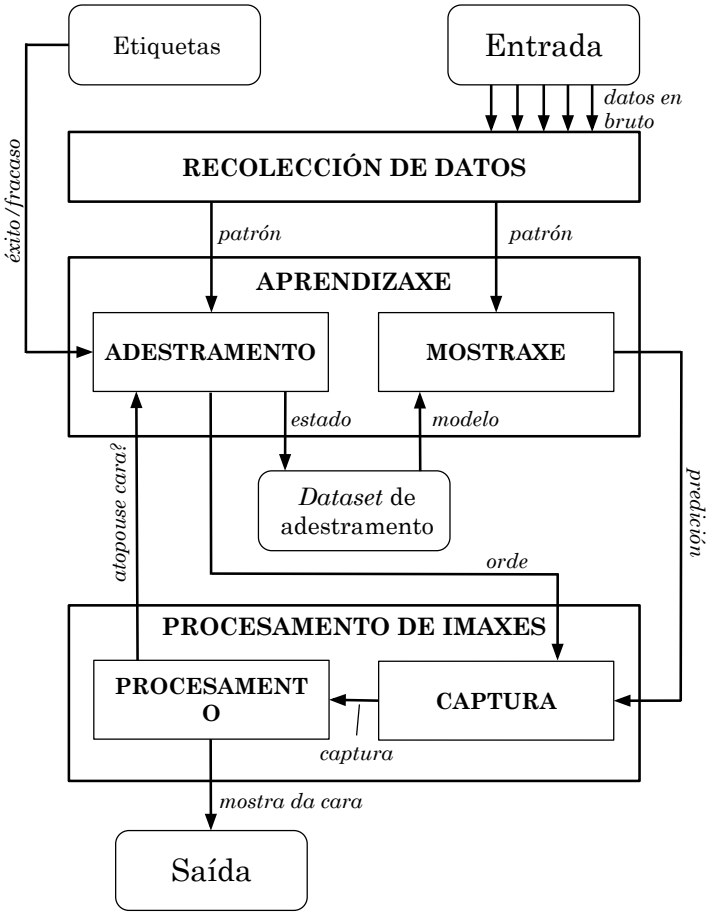


Figura 4.1: Representación da proposta de toma de decisións.

As imaxes tomadas polo dispositivo pertencerán a unha das seguintes clases:

- **Éxito**: unha imaxe na que se recoñece unha cara coa suficiente calidade (resolución, iluminación, ausencia de *blur*, etc).

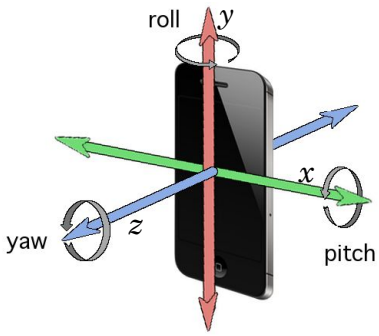


Figura 4.2: Rotacións *pitch*, *roll* e *yaw* sobre un *smartphone*.

- **Fracaso**: unha imaxe que non cumpre algún dos requisitos para ser éxito, ben sexa que non se atopa ningunha cara, detéctanse varias ou a calidade non é a esperada.

Estamos a falar dun problema de **aprendizaxe supervisada**[1]. Máis concretamente, trátase dun problema de clasificación, onde se debe etiquetar a cada estado coma un éxito ou un fracaso.

O módulo de **Recolección de datos** é o responsable de obter o conxunto de datos de entrada e procesalo para xerar o patrón de aprendizaxe, representado por un vector coa información do contexto. Este patrón é a entrada principal do módulo de **Aprendizaxe**.

A aprendizaxe divídese en dúas etapas. O que se debe facer na primeira fase, de **adestramento**, é crear un conxunto de datos de adestramento formado polo total de estados capturados e clasificados cunha **etiqueta** coma un éxito ou un fracaso. O módulo de **Procesamento de imaxes** tomará unha foto por cada estado entrante para determinar a súa clase. Será necesario procesar cada imaxe para ver se existen rostros na mesma.

Unha vez que se dispón dun *dataset* de adestramento, pásase á segunda fase, de **mostraxe**, na que se tomarán datos do contexto de forma periódica e se contrastarán cos obtidos na fase anterior co fin de predicir se se dan as condicións axeitadas para facer unha nova captura e obter un éxito. Unha vez que se decide tomar a imaxe, debe ser procesada para obter retroalimentación acerca do éxito ou fracaso da decisión.

[1]En aprendizaxe automática e minería de datos, a aprendizaxe supervisada é unha técnica para deducir una función a partir de datos de adestramento. Os datos de adestramento consisten de pares de obxectos (normalmente vectores): unha compoñente do par son os datos de entrada e a outra, os resultados desexados. O obxectivo é poder predicir o valor correspondente a calquera obxecto de entrada válida despois de ver unha serie de exemplos, os datos de adestramento.

4.1.1. Obtención do patrón

O patrón que nos proporciona o estado do dispositivo inclúe información relativa á súa pose, manipulación e iluminación do entorno. Vexamos en detalle como se obtén cada compoñente.

Cálculo da pose

Necesitamos coñecer a pose do teléfono móbil (*pitch*, *yaw*) respecto dun marco de referencia inercial para poder determinar o estado do mesmo. Este marco inercial é un sistema ortogonal composto de tres eixes: polo norte magnético, este terrestre e centro da Terra.

Para proxectar as lecturas provintes dos sensores do sistema de referencia local do móbil (Figura 4.2) no sistema inercial que acabamos de mencionar, empregamos unha representación mediante un **cuaternión**. Un cuaternión ($\boldsymbol{q}$) é un vector de catro dimensións que representa a transformación entre dous sistemas de referencia como a rotación dun ángulo θ en torno ao eixe $\boldsymbol{u} = [u_x\ u_y\ u_z]$. Polo tanto:

$$\boldsymbol{q} = \begin{bmatrix} cos\theta \\ sin\theta\ u_x \\ sin\theta\ u_y \\ sin\theta\ u_z \end{bmatrix} \tag{4.1}$$

Seguindo o **método de Madgwick** [20, 21], a través da información provista polo **xiroscopio** e o **acelerómetro** do dispositivo, podemos obter un cuaternión que represente as transformacións para pasar do sistema local do móbil ao inercial da Terra.

O xiroscopio mide a velocidade angular ao redor dos eixes x, y, e z do sistema local, denominados ω_x, ω_y e ω_z respectivamente. Se estes parámetros son dispostos no vector de catro dimensións ${}^{S}\boldsymbol{\omega}$, definido na Ecuación (4.2), a derivada do cuaternión que describe a velocidade de cambio da orientación do marco terrestre respecto do marco do sensor ${}^{S}_{E}\dot{\boldsymbol{q}}$ pode ser calculada empregando a Ecuación (4.3). Polo tanto, a orientación do sistema inercial da terra respecto do local no momento t, ${}^{S}_{E}\boldsymbol{q}_{\omega,t}$, pode ser calculado integrando a través do tempo a derivada do cuaternión, tal e como se describe na Ecuación (4.4), onde ${}^{S}\boldsymbol{\omega}_t$ é a velocidade angular medida no momento t, Δt é o período de obtención da información e ${}^{S}_{E}\widehat{\boldsymbol{q}}_{est,t-1}$ é a estimación previa da orientación.

$${}^{S}\boldsymbol{\omega} = [0\ \omega_x\ \omega_y\ \omega_z] \tag{4.2}$$

$$ {}^{S}_{E}\dot{\boldsymbol{q}} = \frac{1}{2} {}^{S}_{E}\widehat{\mathbf{q}} \otimes {}^{S}\boldsymbol{\omega} \tag{4.3} $$

$$ {}^{S}_{E}\boldsymbol{q}_{\omega,t} = {}^{S}_{E}\widehat{\boldsymbol{q}}_{est,t-1} + \left(\frac{1}{2} {}^{S}_{E}\widehat{\boldsymbol{q}}_{est,t-1} \otimes {}^{S}\boldsymbol{\omega}_{t} \right) \times \Delta t \tag{4.4} $$

No referente ao acelerómetro, este sensor mide as aceleracións que experimenta o móbil nos tres eixes do sistema de referencia local: a_x, a_y and a_z. De novo, estes valores poden ser dispostos nun vector, ${}^{S}\widehat{\boldsymbol{a}}$, como amosa a Ecuación (4.5). Cando o móbil non está sometido a ningunha outra forza externa que non sexa a gravidade, a única aceleración que o dispositivo debe experimentar é aquela da gravidade, isto é [0 0 9,8] m/s^2, de acordo co sistema de referencia inercial. Polo tanto, a proxección do vector de catro dimensións normalizado ${}^{S}\widehat{\boldsymbol{a}}$ no sistema de referencia inercial, ${}^{E}\widehat{\boldsymbol{g}}$, debe coincidir coa Ecuación (4.6). Por conseguinte, debería suceder que ${}^{S}_{E}\widehat{\boldsymbol{q}}^{*} \otimes {}^{E}\widehat{\boldsymbol{g}} \otimes {}^{S}_{E}\widehat{\boldsymbol{q}} = {}^{S}\widehat{\boldsymbol{a}}$. Esta é a razón pola cal o cuaternión será aquel que minimice a diferenza da Ecuación (4.7).

$$ {}^{S}\widehat{\boldsymbol{a}} = [0 \; a_x \; a_y \; a_z] \tag{4.5} $$

$$ {}^{E}\widehat{\boldsymbol{g}} = [0 \; 0 \; 0 \; 1] \tag{4.6} $$

$$ {}^{S}_{E}\widehat{\boldsymbol{q}} = \underset{{}^{S}_{E}\widehat{\boldsymbol{q}} \in \mathbb{R}^4}{\arg\min} \left({}^{S}_{E}\widehat{\boldsymbol{q}}^{*} \otimes {}^{E}\widehat{\boldsymbol{g}} \otimes {}^{S}_{E}\widehat{\boldsymbol{q}} - {}^{S}\widehat{\boldsymbol{a}} \right) \tag{4.7} $$

Finalmente, podemos combinar as dúas fontes de información, Ecuacións (4.4) e (4.7), para acadar o cuaternión en cada instante, tal e como se amosa na Ecuación (4.8), onde f provén da Ecuación (4.7), tal que $f = {}^{S}_{E}\widehat{\boldsymbol{q}}^{*} \otimes {}^{E}\widehat{\boldsymbol{g}} \otimes {}^{S}_{E}\widehat{\boldsymbol{q}} - {}^{S}\widehat{\boldsymbol{a}}$.

$$ {}^{S}_{E}\widehat{\mathbf{q}}_{est,t} = {}^{S}_{E}\widehat{\mathbf{q}}_{est,t-1} + \gamma \left(-\mu \frac{\nabla f}{||\nabla f||} \right) + (1-\gamma) \left(\frac{1}{2} {}^{S}_{E}\widehat{\mathbf{q}}_{est,t-1} \otimes \; {}^{S}\boldsymbol{\omega}_{t} \right) \times \Delta t \tag{4.8} $$

Iluminación, toque de pantalla e movemento

Evidentemente, o módulo da aceleración tamén emprega os datos do acelerómetro para ser calculado. Valores próximos a 9,8 corresponderanse cun estado de repouso no dispositivo. A información lumínica é capturada mediante o **sensor de luz** do dispositivo e a súa unidade de medida é o lux (lx). A ausencia de luz corresponde a 0 lx. Por último, o toque de pantalla non se obtén a través

de ningún sensor, senón mediante un **servizo Android** expresamente creado e personalizado para ese obxectivo, que se executa en segundo plano detectando as pulsacións. Calcúlase a inversa do tempo transcorrido desde o último toque para evitar que o valor almacenado medre de xeito incontrolado.

Actualmente, o toque de pantalla está sempre deshabilitado na máscara do patrón, é dicir, estase capturando pero non se aplica na aprendizaxe. Isto é porque co adestramento en primeiro plano é imposible simular unha interacción natural entre o usuario e a pantalla do dispositivo de cara a que esta compoñente aporte valor na fase de mostraxe.

4.1.2. Algoritmos de clasificación

FaceSampler está pensada para permitir integrar e probar novos algoritmos de forma doada. Ata a data, fóronse introducindo de forma progresiva os que se comentan a continuación.

Always true

Trátase dun algoritmo carente de capacidade preditiva. É o máis sinxelo de todos os implementados. En ningún caso é un algoritmo real de aprendizaxe supervisada, pero simúlao, xa que de cara ao exterior require dos mesmos parámetros que as outras alternativas. Pódese considerar un algoritmo "optimista", xa que cando se lle solicita unha predición, sempre considera que se trata dun bo momento para tomar unha captura e obter éxito.

Pese á súa simplicidade, *always true* ten dúas aplicacións relevantes. En primeiro lugar, é o algoritmo empregado cando un dispositivo é incapaz de obter información imprescindible do contexto para definir os patróns. Por exemplo, aínda que non sexa habitual, hai modelos concretos que carecen de sensores como o acelerómetro ou o xiroscopio, fundamentais para determinar a pose do dispositivo. Por outro lado, resultou unha boa ferramenta de depuración durante a efectuación daquelas probas de caixa negra nas que se precisaba de capturar o maior número de imaxes ou coa maior velocidade posible.

k-NN

O método dos k veciños máis próximos (do inglés *k-nearest neighbors*, abreviado k-NN) [9, Chapter 4] é o algoritmo empregado por defecto en FaceSampler. Na fase de mostraxe, de forma periódica obtense o estado do dispositivo e aplícase o k-NN para saber se tomar a foto ou non en función dos datos de adestramento que se

almacenaron previamente. Nunha estratexia inicial procurábanse os $k = 5$ veciños máis próximos, é dicir, os cinco estados aprendidos que máis se semellasen ao actual, e tomábase a foto se polo menos a metade dos veciños estaban etiquetados como éxito. Esta estratexia era demasiado pouco restritiva, polo que se decidiu empregar $k = 3$, sendo a condición de tomar a foto que os tres veciños tivesen caso de éxito. O resultado con esta estratexia é moito mellor, posto que, aínda que o dispositivo se volve moito máis cauto, a taxa de acertos aumenta notoriamente. Claro está, dita taxa de éxito depende directamente da calidade do adestramento previo.

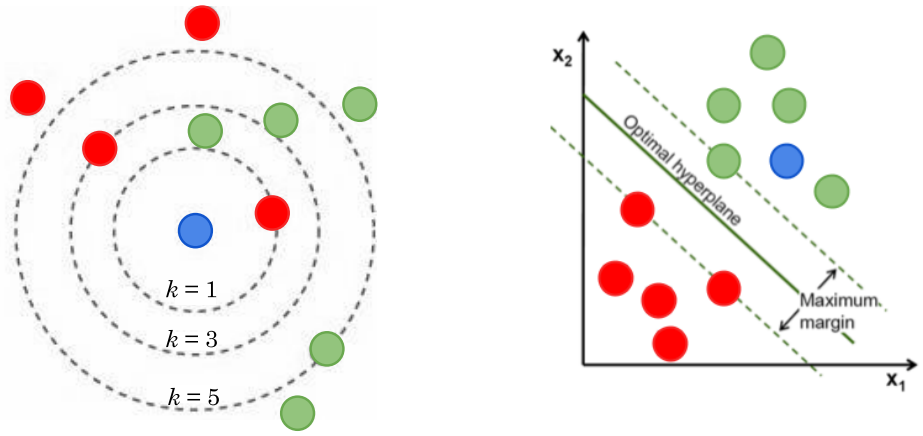
SVM

As máquinas de soporte vectorial (*Support Vector Machines*, SVMs) [9, Chapter 9] son un conxunto de algoritmos de aprendizaxe supervisada orixinariamente ideados para construír un clasificador binario óptimo. As SVMs representan o modelo provinte da fase de adestramento no espazo, separando as clases en dúas rexións o máis amplas posibles mediante un hiperplano de separación definido como o vector entre os dous puntos, das dúas clases, máis próximos ao que se coñece como vector soporte. Cando os novos estados se contrastan con dito modelo, en función da rexión á que pertenzan, poden ser etiquetados cunha ou outra clase. Neste caso, posto que OpenCV permite traballar con SVMs, decidiuse implementar un preditor SVM coa axuda de dita biblioteca.

Na Figura 4.3 amósase unha representación simplificada do problema con k-NN (4.3a) e SVM (4.3b). En ambos casos, o punto azul é o estado actual e os demais son aqueles estados aprendidos durante o adestramento máis próximos ao estado actual. Os vermellos son fracasos e os verdes éxitos.

4.2. A xestión eficiente dos recursos

O terceiro dos obxectivos do proxecto `[OBX-3]` tamén compromete á aplicación a traballar de forma eficiente, xestionando os recursos computacionais da mellor forma posible, minimizando o consumo de enerxía e evitando monopolizar os recursos de procesamento. Para acadar estas metas, durante a evolución do proxecto procurouse en todo momento manter un bo control sobre o traballo realizado pola aplicación, xestionando os **procesos e sub-procesos** involucrados. Do mesmo xeito, de cara á fase de mostraxe, posto que se efectúa durante tempo prolongado (horas, días e incluso semanas), definiuse un **sistema de execución en dúas modalidades** controlado por **eventos disparadores**.



(a) Estratexia k-NN. **(b)** Estratexia SVM.

Figura 4.3: Comparativa entre os dous algoritmos principais.

4.2.1. Procesos e sub-procesos

De maneira predeterminada, todos os compoñentes da mesma aplicación se executan no mesmo proceso e, por norma xeral, non se recomenda modificar isto [22]. Cando unha aplicación Android se inicia, o sistema crea un *thread* (sub-proceso) de execución para a mesma, que se denomina "principal". Este *thread* é moi importante porque, entre outras cousas, é aquel sobre o cal a aplicación interactúa cos compoñentes do kit de ferramentas da interface de usuario (IU) de Android. Por iso, ao *thread* principal tamén se lle coñece como o **sub-proceso de IU**.

Sen embargo, cando a aplicación realiza traballo intensivo en resposta a unha interacción do usuario, este modelo de *thread* único pode xerar un rendemento deficiente. Especificamente, se todo sucede no sub-proceso de IU, realizar operacións prolongadas, como é o caso do procesado continuo das imaxes e os patróns en FaceSampler, bloqueará toda a IU. Desde a perspectiva do usuario, a aplicación non responderá. O que é peor, se o sub-proceso de IU está bloqueado por máis duns poucos segundos, ao usuario presentaráselle un cadro de diálogo de tipo ANR ("a aplicación no responde").

Para evitar reportes de erros ANR, adicouse tempo a realizar unha xestión eficiente dos procesos e sub-procesos da aplicación. Tan só se require do *thread* principal para o control da interface gráfica e de outro máis para executar a mostraxe, ao cal lle chamaremos **sub-proceso de mostraxe**. Este segundo é imprescindible para executar as tarefas de captación do estado do dispositivo, captura de imaxes, detección de caras, etc., sen que o rendemento da IU se vexa afectado. Na etapa de mostraxe, o usuario pode seguir navegando por FaceSampler ou o resto de

aplicacións do seu dispositivo, e en ningún caso se pode tolerar que o rendemento do mesmo se vexa prexudicado por unha mala xestión dos recursos.

No caso da etapa de adestramento, en principio non é preciso outro *thread* xa que todo o traballo se realiza en primeiro plano, reflectindo todo o que sucede na IU. Non obstante, en determinadas ocasións FaceSampler necesita realizar algunha actualización da IU con información provinte das saídas de tarefas pesadas, que poden retardar a velocidade de resposta da IU. O problema é que interface gráfica de Android non permite chamadas desde outros *threads* que non sexan o seu, así que se intentamos crear un novo *thread* con este obxectivo, a aplicación finalizará. Unha boa solución para este tipo de situacións provista por defecto polo entorno Android é recorrer ao emprego de **AsyncTasks** (`android.os.AsyncTask<Params, Progress, Result>`) [23]. Unha AsyncTask permite realizar traballo asíncrono na interface de usuario. Leva a cabo as operacións de bloqueo nun sub-proceso de traballo e logo publica os resultados no sub-proceso de IU, sen necesidade de que o programador teña que manexar subprocesos ou controladores.

4.2.2. Modalidades de execución e eventos disparadores

A mostraxe é un proceso que pode durar moito tempo, e non convén ocupar recursos como a cámara de xeito prolongado nin estar continuamente analizando o estado do dispositivo cando sabemos que non é un bo momento para tomar unha captura. Por iso, a aplicación diferenza dous modos de execución da mostraxe:

- **Modo normal** (*Normal mode*). O proceso está "alerta" para realizar unha captura. Tómase o control sobre o recurso da cámara. Periodicamente recóllese toda a información actualizada dos sensores e servizos, realízanse os cálculos necesarios para obter o estado (cálculo do cuaternión, módulo da aceleración, proxeccións, etc.) e predise se é un bo momento para obter unha nova mostra. De selo, faise a captura, almacénase na memoria do dispositivo, procésase, trátase de detectar o rostro do usuario e, de ser definitivamente un éxito, almacénase o mesmo na memoria como mostra.
- **Modo aforro de enerxía** (*Low Energy mode*). O proceso está "durmido", que non implica pausado nin detido. Periodicamente tómase a información dos sensores e servizos, pero non se procesa nin se chega a obter o estado. O recurso da cámara está libre. Este modo de execución limítase a agardar que se orixine un evento para volver de novo ao modo normal.

A permutación entre modos realízase en base a unha serie de **regras e eventos** que poderán ir variando en futuras versións da ferramenta. A día de hoxe, son os seguintes:

1. **Bloqueo**. Cando se bloquee o dispositivo (evento `ACTION_SCREEN_OFF` en Android), solicitarase entrar en modo aforro de enerxía. É moi improbable que se den as circunstancias idóneas para tomar unha mostra cando o teléfono está bloqueado; o habitual é que estea gardado no peto do usuario, pousado nunha mesa, etc.

2. **Desbloqueo**. Cando se desbloquee o dispositivo (evento Android `ACTION_USER_PRESENT`), solicitarase pasar a modo normal. Compre remarcar a diferencia entre desbloqueo do dispositivo e acendido da pantalla (evento `ACTION_SCREEN_ON`), xa que o primeiro pode implicar que o usuario acceda ao escritorio do dispositivo mediante un patrón, contrasinal, PIN, etc. Mentres o usuario estea a empregar o móbil, será moi probable obter unha boa mostra del.

3. **Batería baixa**. Cando o dispositivo entre en estado de "batería baixa", o proceso pasará a modo aforro de enerxía para procurar prolongar o tempo de uso do móbil.

4. **Solicitude do usuario**. Permítese ao usuario permutar de xeito manual entre os dous modos, pulsando os botóns de control proporcionados na notificación do proceso adherida á barra de notificacións. Na sección 5.2.2 explícase en detalle esta interacción.

5. **Cámara ocupada**. Se o proceso se atopa en modo aforro de enerxía e o usuario está empregando a cámara do dispositivo para outras finalidades e, de súpeto, se solicita entrar en modo normal, o proceso volverá a durmirse. FaceSampler sempre terá a menor prioridade de acceso á cámara, co fin de non interferir noutras actividades do usuario.

6. **Reinicio do dispositivo**. Cando se reinicie o móbil, o proceso tamén se reiniciará. Actualmente, por complicacións técnicas, non se comproba se a mostraxe estaba en execución cando se apagou o dispositivo, polo que cando este se volva a acender, o proceso será iniciado en modo aforro de enerxía co obxectivo de recordar ao usuario a presencia da ferramenta.

7. **Éxito**. Se se toma unha imaxe que serve como mostra, entón solicitarase entrar en modo aforro de enerxía durante un tempo. Por defecto esperaranse 30 segundos para volver a iniciar o modo normal, pero o usuario poderá modificar esa cantidade desde o panel de preferencias.

8. **Fracaso reiterado**. Se durante a execución en modo normal se decidiu por catro veces, consecutivas ou non, realizar unha captura, fracasando en todas as ocasións, entón éntrase en modo aforro de enerxía durante un tempo, que de novo será configurable polo usuario (30 segundos por defecto). O contador de fracasos reiniciarase ao momento de mandar o proceso a modo aforro de enerxía.

Nas últimas dúas situacións, ao mesmo tempo que se solicita entrar en modo aforro de enerxía tamén se programa unha alarma para volver a modo normal. Esta será lanzada transcorrido o tempo acordado. Evidentemente, se o móbil é bloqueado e desbloqueado antes de que transcorra ese tempo, entrarase de novo no modo normal sen esperar pola alarma.

4.3. As tecnoloxías empregadas

Esta sección recolle as ferramentas e tecnoloxías empregadas durante o proxecto. Farase maior fincapé naquelas relativas ao desenvolvemento, pero tamén se comentarán as empregadas para labores de xestión, co fin de ter todo o material empregado resumido nunha única sección.

4.3.1. Equipo e dispositivo de desenvolvemento

Como equipo de traballo empregouse un ordenador de uso particular. Trátase dun **Lenovo Yoga Pro 3** que conta con sistema operativo Ubuntu 16.04 LTS de 64 bits.

Ademais, como xa se comentaba no capítulo introdutorio, necesitouse dun dispositivo Android para o despregamento da aplicación durante o seu desenvolvemento. O Grupo de Sistemas Intelixentes da USC proporcionou un **BQ Aquaris E5s**, que conta con Android 5.1.1 (Lollipop), así como con unha cámara frontal de 5 MP, Full HD a 30 fps e unha pantalla IPS con Quantum Color de cinco polgadas con resolución de 1.280x720 píxeles.

4.3.2. Ferramentas

Durante o desenvolvemento, empregouse o IDE **Android Studio 2.1.2** para a construción da aplicación. Escolleuse este IDE e non outro por ser o recomendado nas guías de desenvolvemento de Google. Tamén se utilizou **StarUML 2.8.0** como software de modelado UML. Neste caso a selección veu dada en base á familiaridade existente coa ferramenta.

Para a xestión do proxecto, empregouse **Git** por consola de comandos como software de control de versións, tal e como se comentaba na Sección 3.1. A ferramenta empregada para a documentación foi **TeXstudio 2.11.2**. O control da Pila de Produto realizouse con **LibreOffice Calc 5.1.6.2**. Finalmente, utilizouse **ProjectLibre 1.5.7** para o control dos recursos e a planificación temporal.

4.3.3. Linguaxes de programación

Tipicamente, as aplicacións Android desenvólvense na linguaxe Java con Android Software, aínda que tamén se pode traballar noutras linguaxes como C o C++. Neste caso, optouse por empregar **Java** non só por ser o habitual, senón tamén pola experiencia adquirida durante o grao nesta linguaxe.

A maiores, necesítase estar en posesión do SDK de Android, do cal falaremos no seguinte apartado, ademais de certa destreza en **XML**. Este último é un dos estándares máis utilizados na actualidade para codificar información. É amplamente empregado en Internet, e en Android recibe múltiples usos. Entre outras cousas, é utilizado para definir *layouts*, animacións, o manifesto[1], etc.

4.3.4. Dependencias e bibliotecas

Como se comentaba no apartado anterior, para saber desenvolver aplicacións en Android non basta con coñecer as linguaxes Java ou XML. Tamén hai que estar en posesión e saber manexar o kit de desenvolvemento de software, ou SDK, provisto por Google, o cal se pode descargar gratuitamente. O **SDK de Android**, inclúe un conxunto de ferramentas de desenvolvemento que abarcan un depurador de código, a biblioteca, un simulador de teléfono baseado en QEMU, documentación, exemplos de código e tutoriais. As actualizacións do SDK están coordinadas co desenvolvemento xeral de Android. O SDK soporta tamén versións antigas de Android, por si os programadores necesitan instalar aplicacións en dispositivos xa obsoletos ou máis antigos. As ferramentas de desenvolvemento son compoñentes descargables, de modo que una vez instalada a última versión, poden instalarse versións anteriores e facer probas de compatibilidade.

Para o tratamento das imaxes capturadas cumpría dispor dunha biblioteca de visión artificial que permitise, principalmente, detectar rostros nas mesmas. Para esta tarefa elixiuse **OpenCV**, biblioteca libre orixinalmente desenvolta por Intel. Sen lugar a dúbidas trátase da máis coñecida e, segundo as críticas, tamén a máis completa e activa. Conta con máis de 500 algoritmos entre os que se inclúen funcións de propósito xeral para procesamento de imaxes, descricións xeométricas, segmentación, seguimento, etc. Dispón de abundante documentación, incluíndo algúns libros. É multiplataforma (incluíndo, claro está, Android) e soporta as linguaxes de programación C++, C, Python e Java.

[1]Todas as aplicacións deben ter un arquivo `AndroidManifest.xml` (con ese nome exacto) no directorio raíz. O manifesto proporciona información esencial sobre a aplicación ao sistema Android, información que o sistema debe ter para poder executar o código da app [24].

Capítulo 5

Deseño

No presente capítulo preténdese recoller como foi construída FaceSampler, documentando os seus compoñentes e a comunicación entre eles. Partirase dunha visión global cun nivel alto de abstracción ata chegar aos detalles e particularidades de cada compoñente. Comentarase tamén o equipamento hardware e software necesario e xustificarase cada unha das decisións tomadas.

A calidade do produto final depende substancialmente do esforzo aplicado durante a etapa de deseño. Seguindo a filosofía Scrum, o deseño neste proxecto realizouse de maneira continua, e foise actualizando e completando en cada *sprint*. Os detalles dos compoñentes (capas, módulos e clases), e a interacción entre os mesmos foise perfeccionando á par que se realizaba a implementación, nun proceso de mellora constante. Para isto, empregáronse **tarxetas CRC**[1] como ferramenta principal de deseño áxil. As tarxetas CRC foron de gran utilidade durante o desenvolvemento do sistema para documentar o significado de cada clase durante a asignación de responsabilidades, estruturar o conxunto de clases e simular escenarios.

5.1. Arquitectura do sistema

Nesta sección preséntase a estruturación global do sistema. Esta implica un deseño de alto nivel que ten dous propósitos principais: satisfacer os atributos de calidade (desempeño, seguridade, modificabilidade), e servir como guía no desenvolvemento.

[1] A utilización de tarxetas CRC (*Class-Responsibility-Collaboration*) é unha ferramenta de *brainstorming* empregada como metodoloxía para o deseño de software orientado a obxectos, que foi proposta por Kent Beck (introdutor da metodoloxía áxil de programación extrema).

Scrum, como metodoloxía áxil que é, avoga por un modelo de desenvolvemento no que os avances no deseño do sistema se producen de xeito paralelo á propia implementación. Non obstante, o feito de non ter unha arquitectura robusta definida desde etapas temperás do desenvolvemento pode limitar severamente o que o produto final satisfaga as necesidades do cliente. Ademais, o custo das correccións relacionadas con problemas na arquitectura é moi elevado. Por tanto, as decisións críticas relativas ao deseño xeral dun sistema software complexo como este deben de facerse desde un comezo.

Estas consideracións foron tomadas en conta á hora de facer a planificación inicial do proxecto. Deste xeito, na etapa de análise previo ao desenvolvemento, adicouse tempo a obter unha aproximación arquitectónica factible ao mesmo tempo que se comprendía o alcance do proxecto e se seleccionaban as tecnoloxías, algoritmos e metodoloxías a aplicar. Hai que ter en conta que nesta sección se plasma a versión actual e definitiva da arquitectura, pero durante o desenvolvemento do proxecto efectuáronse varias modificacións orixinadas por cambios na especificación dos requisitos.

Na Figura 5.1 preséntase unha primeira aproximación á estrutura do sistema. Trátase dun diagrama *ad-hoc* que inclúe os principais elementos con propiedades visibles de forma externa e as relacións que existen entre eles.

O gran bloque central deste diagrama representa á aplicación, a cal se estrutura en tres capas, seguindo o **patrón arquitectónico MVC** [25] (Modelo Vista Controlador). Compre destacar que MVC é un concepto en lugar dun sólido marco de programación, e non existe un patrón MVC universal. Deste xeito, pódese adaptar MVC para calquera plataforma mentres se siga a idea básica proposta por este patrón, que consiste en ter clara a finalidade de cada unha das tres capas:

- **Modelo**: responde a que fai a aplicación.
- **Vista**: decide como se amosa a información.
- **Controlador**: xestiona eventos e a interacción do usuario.

Tendo isto en conta, en FaceSampler empregouse unha adaptación propia de MVC que pretende dar solución ao problema concreto e se adapta á plataforma Android. Os principais matices introducidos son os seguintes:

1. A interface de usuario (Vista) defínese, principalmente, por medio de arquivos XML, aínda que tamén haberá determinados elementos que se xeren de forma dinámica en Java en tempo de execución.

2. Empréganse clases que estenden de `android.app.Activity` para "inflar"[1] os recursos XML da Vista e facer uso deles, xestionando a interacción do

[1] Inflar unha vista en Android (*inflate*) supón instanciar un recurso de *layout* XML en tempo de execución desde unha Activity para poder engadilo a unha xerarquía de vistas [26, 27].

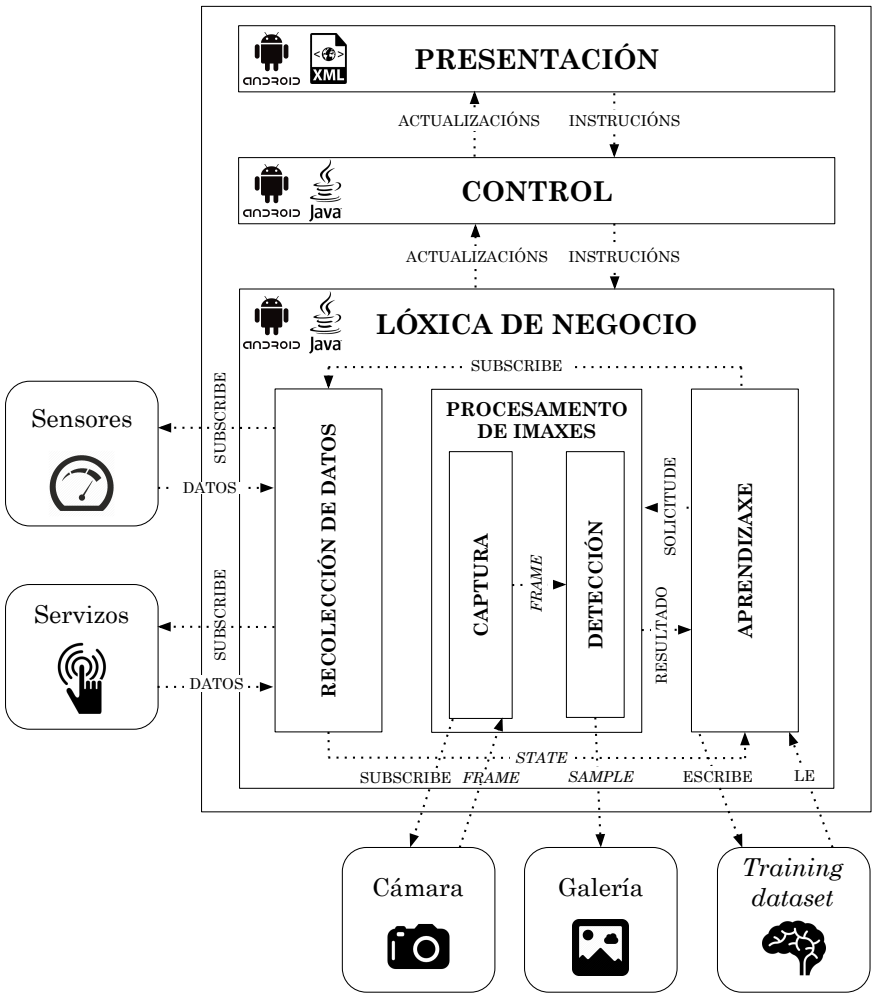


Figura 5.1: Diagrama *ad-hoc* da arquitectura lóxica do sistema.

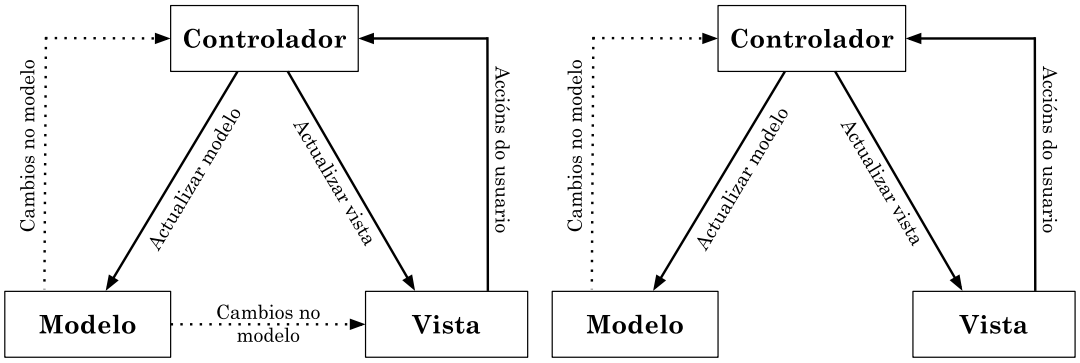
usuario cos mesmos (Controlador).

3. A lóxica de negocio modélase mediante clases Java tradicionais (Modelo).
4. Non pode existir interacción directa entre a Vista e o Modelo.

A Figura 5.2 compara o patrón clásico (5.2a) coa adaptación realizada para este contexto (5.2b).

Volvendo á Figura 5.1, podemos comprobar como este diagrama segue o patrón 5.2b. Neste caso, a capa de **Presentación** é a Vista, a capa de **Control** é o Controlador e a capa de **Lóxica de Negocio** é o Modelo, que á súa vez se estrutura en tres módulos diferenciados: Recolección de datos, Procesamento de Imaxes e Aprendizaxe.

O módulo de **Recolección de datos** é o encargado de obter a información do



(a) Patrón estándar. **(b)** Adaptación para este contexto.

Figura 5.2: Comparativa entre dúas propostas MVC.

entorno e proporcionar o estado [RI-6] do dispositivo cando este sexa solicitado. Por un lado, subscríbese aos sensores do dispositivo (acelerómetro, xiroscopio, magnetómetro e sensor de luz) para que estes lle proporcionen información continua do entorno [RI-3]. Ademais, tamén accede a outra información provinte de servizos [RI-4], que os sensores non poden proporcionar.

O módulo de **Procesamento de Imaxes** leva a cabo tres tarefas. En primeiro lugar, toma capturas [RI-1] baixo demanda a través da cámara frontal do dispositivo. En segundo lugar, procesa esas capturas co obxectivo de detectar o rostro do usuario nas mesmas para determinar se poden ser boas mostras [RI-2] para o banco de caras. Finalmente, almacena as capturas e mostras na memoria interna do móbil.

O módulo de **Aprendizaxe** encárgase de dotar ao sistema da intelixencia necesaria para a toma de decisións respecto de realizar capturas. Para elo, apóiase na información histórica que el mesmo garda na base de coñecemento [RI-7], así coma nos datos en tempo real provintes do módulo de Recolección de datos. Deste xeito, cando considere que é un bo momento para obter unha mostra comunicarallo á capa de Control para que esta lle solicite unha captura ao módulo de Procesamento de Imaxes.

Convén remarcar que a Figura 5.1 amosa un deseño simplificado de xeito intencionado, co obxectivo de ser unha boa primeira aproximación á estruturación do sistema. Non obstante, en ningún caso pretende ser unha representación fidedigna da arquitectura definitiva. Por exemplo, a interacción que se amosa entre os distintos módulos da capa de Lóxica non é directa, senón que existe un intermediario, o controlador, que é quen realmente realiza as subscricións e obtén os resultados.

Comprendida a filosofía global do sistema, podemos introducir agora outra re-

presentación da arquitectura, esta vez proporcionando un maior detalle técnico. A Figura 5.3 amosa o diagrama de compoñentes UML da aplicación. Vemos que todos os nomes de compoñentes, estereotipos e incluso anotacións están en inglés. O diagrama *ad-hoc* anterior amosouse en galego por tratarse dunha representación non formal a alto nivel pero, a partir de agora, todo o modelado do sistema se fará en inglés, posto que se elixiu ese idioma tamén para a codificación por ser o universal. Os nomes dos compoñentes da Figura 5.3 coinciden cos outorgados no proxecto Android Studio aos paquetes e directorios asociados a cada un dos mesmos.

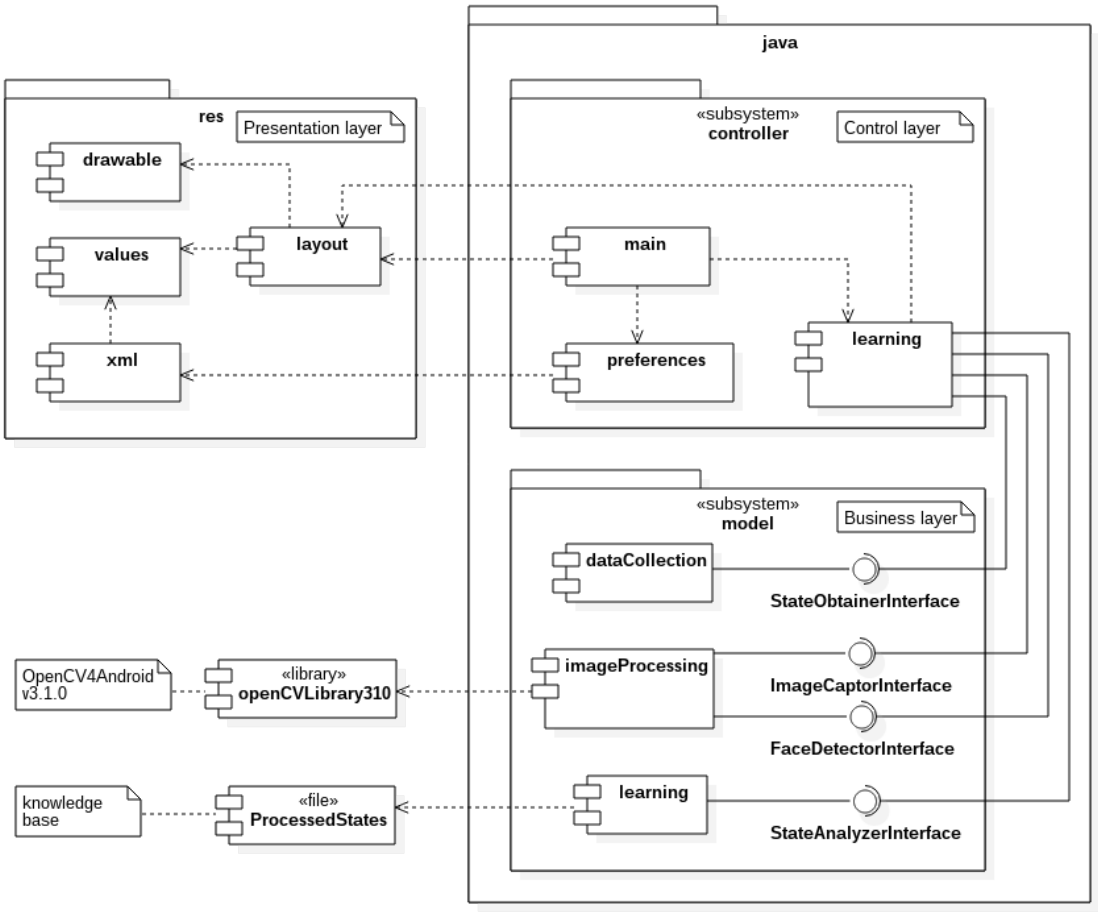


Figura 5.3: Diagrama de compoñentes UML da arquitectura do sistema.

Neste diagrama, a capa de Presentación coincide co conxunto de recursos Android (`res`), do cal falaremos en detalle na Sección 5.2.2. As capas de Control (`controller`) e de Lóxica de negocio (`model`) inclúense dentro do ámbito Java do proxecto. Importante observar como a capa de Lóxica lle proporciona catro interfaces diferenciadas á capa de Control, as cales serán comentadas na Sección 5.4.

O diagrama adianta algúns matices máis que serán desenvoltos en seccións posteriores, como o feito de que o módulo de Procesamento de Imaxes se apoie na biblioteca OpenCV ou que a base de coñecemento utilizada polo módulo de Aprendizaxe sexa un único arquivo.

Ata aquí chega o resumo do funcionamento xeral do sistema. Nas seguintes seccións presentarase o deseño detallado de cada unha das capas.

5.2. Deseño da capa de presentación

A capa de presentación é aquela que lle presenta o sistema ao usuario, lle comunica a información e captura as instrucións do mesmo. Tamén nos podemos referir a ela directamente como interface gráfica ou de usuario. Nesta sección recóllese a estratexia seguida para o seu deseño, así como os resultados obtidos. Na nosa arquitectura, esta capa comunícase exclusivamente coa capa de control, da que falaremos tamén neste capítulo.

5.2.1. Estratexia de deseño

O deseño en Android está baseado nunha pulcritude brillante na composición da interface. Cada gráfico, botón e texto está acompañado pola idea da limpeza visual pero, ao mesmo tempo, sorprende con pequenos detalles.

Roboto, a tipografía propia deste sistema operativo, é en gran parte o seu aceno de identidade e combínase cun estilo de botóns e cores ben definido. Android avoga pola simplicidade, controlada pero non aburrida, que, en ocasións, rompe os seus propios formalismos para encantar ao usuario.

En FaceSampler preténdese seguir as pautas xerais de Android, polo que a intención foi deseñar unha **interface nativa**, baseada en elementos que veñen preestablecidos na plataforma, pero que contase ao mesmo tempo cunha mínima capa de personalización que brindase un toque de identidade propia á aplicación.

A ferramenta está dotada de intelixencia, integrando para elo toda unha lóxica de automatización, resposta a eventos do entorno e toma de decisións notable, que lle outorga autonomía e independencia do usuario. Isto fai que FaceSampler se poida caracterizar en gran medida por necesitar dunha interacción mínima coa pantalla por parte do usuario. É por iso polo que o deseño se centrou tamén en lograr unha **facilidade de uso extrema**, baseada nun deseño minimalista, que require de moi pouca navegación entre pantallas e brinda escasa interacción de xeito intencionado para fomentar a despreocupación por parte do usuario da ferramenta.

Por suposto, preténdese que a aplicación sexa **agradable esteticamente**. Para elo, óptase por seguir as pautas Material Design[1], respectando os principios de estruturación, dimensionado e proporcionalidade das *layouts*, así como as recomendacións referentes ás iconas (todas as empregadas cumpren os criterios para considerarse estándar) e ás cores (empréganse cores da propia paleta suxerida por Google) [29].

Esta interface ten en conta tamén a **adaptabilidade** a variedade de tamaños e resolucións de pantalla. Para facilitar o deseño, Android clasifica as pantallas en catro grupos para o tamaño (*pequeno*, *normal*, *grande* e *extragrande*) e outros seis para a densidade de píxeles (*ldpi*, *mdpi*, *hdpi*, *xhdpi*, *xxhdpi* e *xxxhdpi*). Todas as iconas empregadas en FaceSampler dispoñen dunha versión distinta para cada unha das seis densidades posibles. O propio sistema Android realiza un axuste automático que clasifica o dispositivo no grupo máis apropiado, seleccionando as iconas correspondentes.

Na Figura 5.4 amósase o primeiro bosquexo da interface, realizado inicialmente a man alzada e posteriormente volto a facer a limpo para integralo nesta memoria. Nel indícanse os compoñentes de cada pantalla, así como o fluxo entre as mesmas. O máis destacable aquí pode ser o feito de que nunha viñeta tan pequena se logra recoller a maior parte da navegación da aplicación.

5.2.2. Interface final

Vista a estratexia de deseño podemos agora comentar a interface gráfica resultado. Aínda que algúns compoñentes da mesma sexan xerados de xeito dinámico en Java, a maioría recóllense en arquivos XML coñecidos como recursos (*resources*). Todos os recursos XML mencionados nesta sección localízanse baixo a ruta `app/src/main/res` do proxecto Android Studio.

A Figura 5.5 amosa a vista principal da aplicación. Esta ofrece as dúas funcionalidades principais, adestramento e mostraxe, e permite cambiar entre elas seleccionando a pestana desexada. Os elementos gráficos comúns da vista principal están descritos no arquivo `[...]/layout-v21/main_view.xml`, de tipo *layout*. A modalidade de mostraxe (5.5b) é a primeira que o usuario ve ao iniciar FaceSampler, e o seu deseño recóllese no recurso `[...]/layout-v21/sampling_view.xml`. Ao premer no botón *Start!*[2] nesta modalidade comezará o proceso de captación de mostras do usuario en segundo plano.

[1] Material Design é un concepto, unha filosofía, unhas pautas enfocadas ao deseño utilizado en Android, pero tamén na web e en calquera plataforma [28]. Recibe o seu nome por estar baseado en obxectos materiais, pezas colocadas nun espazo e nun momento determinado.

[2] Para a memoria optouse por empregar as vistas en inglés, pero hai que ter en conta que o texto incluído nas mesmas variará en función da linguaxe do sistema.

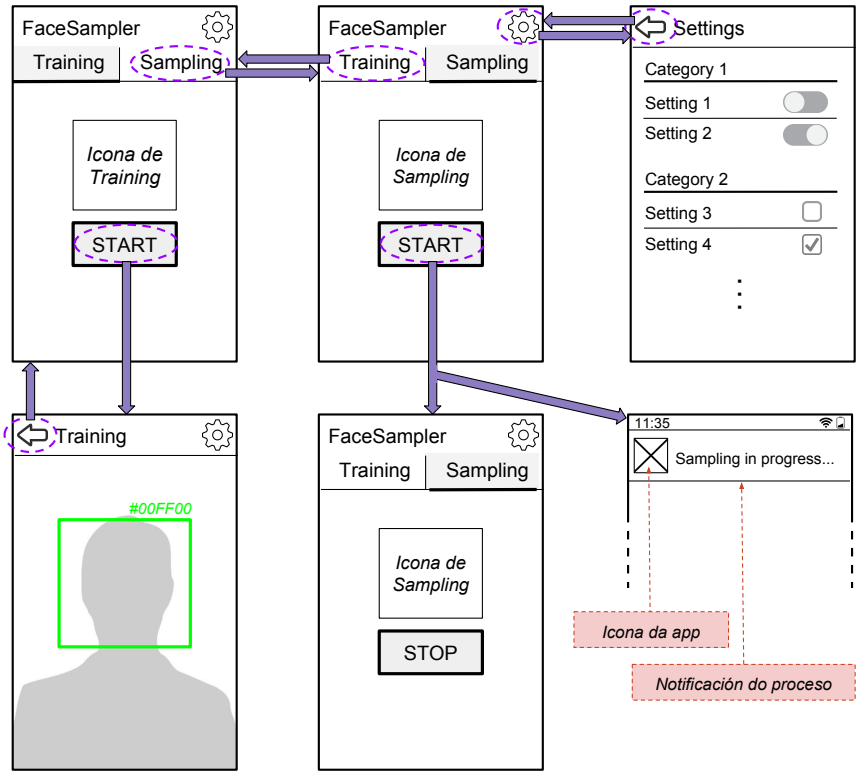
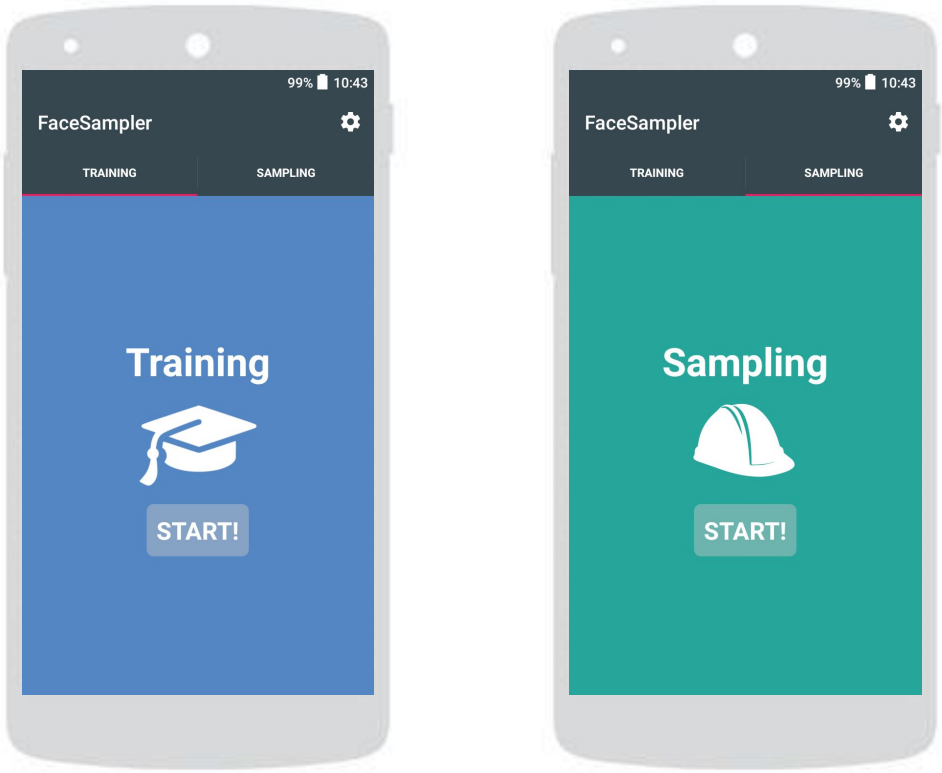


Figura 5.4: Bosquexo inicial da interface gráfica de FaceSampler.

A modalidade de adestramento (5.5a) emprega o deseño descrito no arquivo `[...]/layout/training_view.xml`. Premendo no botón *Start!* deste modo rediríxese ao usuario á vista de adestramento en primeiro plano, a cal se pode ver na Figura 5.6. Nesta pantalla amosaranse en tempo real as imaxes capturadas pola cámara frontal do dispositivo, así como datos interesantes para a aprendizaxe (iluminación, aceleración, orientación do dispositivo, etc.). Cando se detecte o rostro do usuario, marcarase cun cadro verde tal e como amosa a Figura 5.6b.

Sexa cal sexa o estado da vista principal (Figura 5.5), premendo na icona con forma de engrenaxe situada na esquina superior dereita da pantalla estaremos accedendo ao panel de preferencias (Figura 5.7). Desde este poderanse realizar todos os axustes que se desexe, así como consultar a información da versión ou acceder á páxina web da ferramenta. A descrición XML das preferencias recóllese no arquivo `[...]/xml/preferences.xml`.

O panel de preferencias non dispón dunha pantalla única. Existe información que é amosada nunha sub-vista á parte co obxectivo de evitar dispor dun panel único e monolítico no que custe acceder aos contidos. A Figura 5.7b ubícase na sub-vista de *Timings*, e recolle a acción de modificar un valor numérico por medio



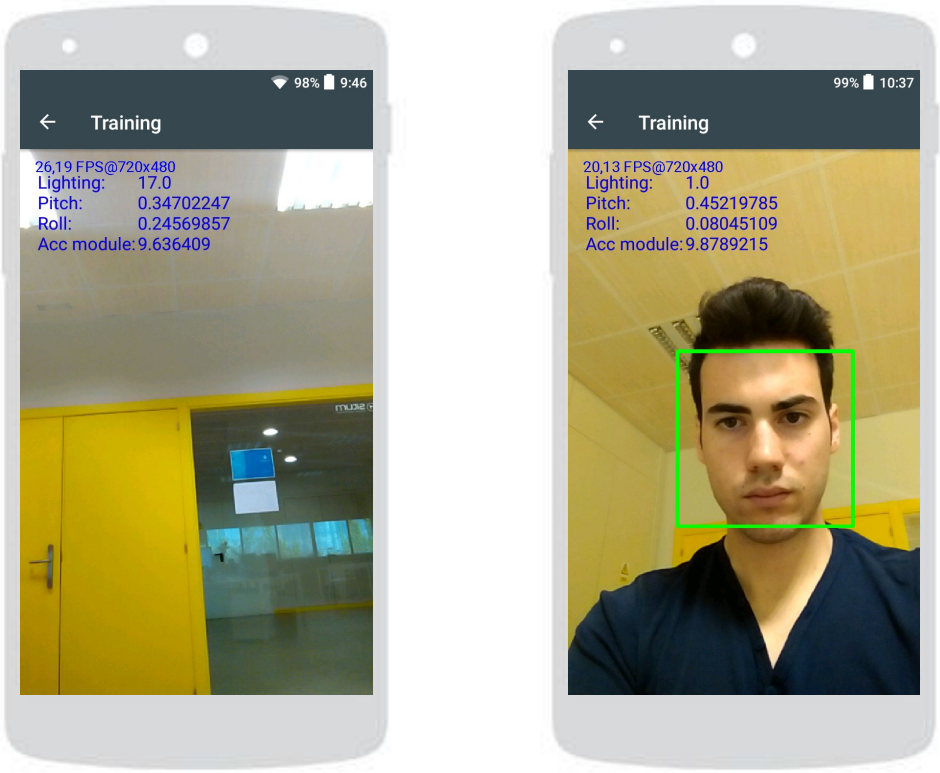
(a) Modo adestramento. **(b)** Modo mostraxe.

Figura 5.5: Vista principal de FaceSampler.

dun diálogo flotante de tipo *number picker*[1]. O deseño deste diálogo atópase no arquivo `[...]/layout/preference_number_picker_dialog.xml`.

Cando se ordena comezar a mostraxe desde a vista principal (Figura 5.5b), lánzase un proceso en segundo plano que nunca se deterá ata que o usuario prema expresamente no botón *Stop!* que substituirá ao de *Start!* amosado antes do inicio. Para que o usuario sexa consciente da existencia de dito proceso, adhírese unha notificación á barra de notificacións de Android tal e como se amosa na Figura 5.8. Aínda que o seu obxectivo principal sexa o de servir como recordatorio, a notificación tamén se emprega como acceso directo á vista principal da aplicación e permite durmir/despertar (*Sleep*/*Wake*) o proceso de mostraxe, así como detelo totalmente (*Stop*).

[1] As preferencias do tipo *number picker* son aquelas que permiten axustar un valor de tipo numérico enteiro (por exemplo, segundos, minutos, cantidades, etc.) por medio dunha roda deslizable.



(a) O usuario non está presente. **(b)** Detéctase un rostro.

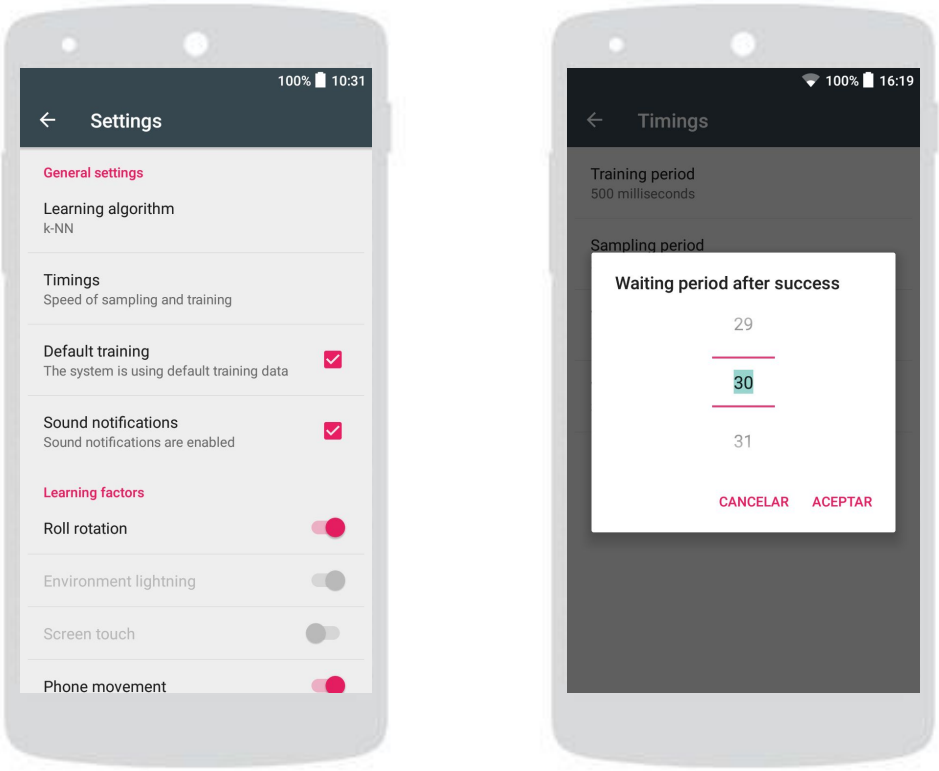
Figura 5.6: Vista de adestramento en primeiro plano.

Icona de lanzamento

Se pensamos na aplicación como un produto que estará nun escaparate xunto a moitos outros, entón a icona de lanzamento é o embalaxe que o envolve. En primeiro lugar, esta icona será símbolo de identidade da aplicación e permitirá identificala nas tendas de aplicacións como elemento de venta para convencer ao usuario de descargala. Sorteado este paso e unha vez instalada no dispositivo, esta terá que convivir con moitas outras das que dispoña o usuario.

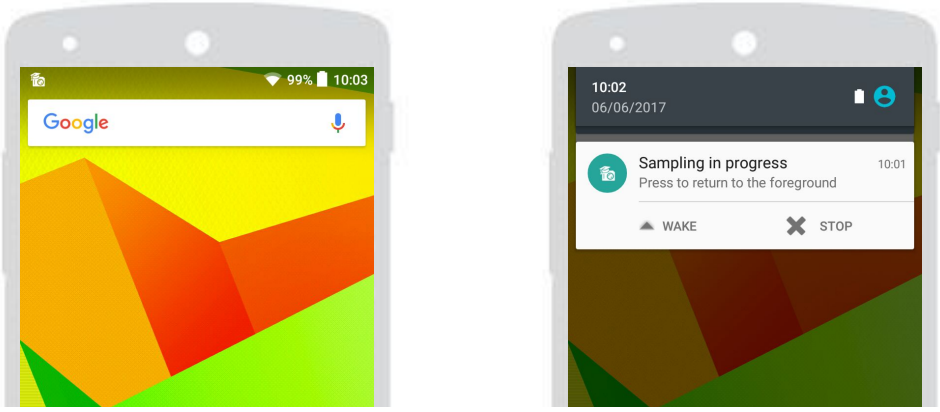
Tendo en conta a súa relevancia, para FaceSampler decidiuse por empeño en crear unha icona de lanzamento distintiva e representativa. Avogouse por formas simples, non moi cargadas e coidadas nos seus detalles. No estándar de Android as iconas son obxectos representados frontalmente, que soen xogar coas sombras, lixeiras perspectivas ou transparencias para non caer na monotonía.

A Figura 5.9 amosa a icona de lanzamento elaborada. O deseño está inspirado na idea de "mostraxe intelixente", onde a cámara simboliza a mostraxe e o birrete a intelixencia.



(a) Pantalla principal. **(b)** Diálogo flotante do tipo *number picker*.

Figura 5.7: Panel de preferencias.



(a) Icona adherida á barra de notificacións. **(b)** Detalle da notificación.

Figura 5.8: Notificación de mostraxe en execución.

Figura 5.9: Icona de lanzamento de FaceSampler.

5.2.3. Análise heurística de usabilidade

Nesta sección recóllese a avaliación heurística da interface de usuario realizada previa implementación da mesma na capa de presentación. Como criterios de avaliación empregáronse os 10 principios heurísticos de Nielsen [30]. O obxectivo desta análise foi detectar errores de usabilidade.

Avaliación

A avaliación consiste en tratar de demostrar que o deseño da interface gráfica cumpre con cada un dos principios de Nielsen, argumentando para cada un deles as características da aplicación que conseguen ditos obxectivos.

1. **Visibilidade do estado do sistema**. A aplicación indica en todo momento na barra de acción en que sección se atopa o usuario. Se se produce algún erro inesperado durante a execución, o sistema Android encargarase de notificalo. Os procesos en execución que non estean á vista do usuario notifícanse por medio de alertas adheridas á barra de notificacións do dispositivo.

2. **Utilización da linguaxe dos usuarios**. A aplicación emprega unha linguaxe natural e nunca termos abstractos. Ademais, sopórtanse tres linguaxes distintas: inglés (por defecto), español e galego. Por último, empréganse as iconas estandarizadas para a plataforma Android, que son perfectamente identificables polos usuarios deste sistema operativo.

3. **Control e liberdade para o usuario**. O usuario coñecerá en todo momento o estado da aplicación grazas ao reforzo visual. Como se comentaba antes, se existe algún proceso en execución que non estea á vista o usuario será notificado. Se o usuario desexa reverter unha acción que realizou por

error, a aplicación volverá á vista ou estado anterior de xeito practicamente instantáneo, aínda que algúns efectos tal vez non poidan ser reversibles (como re-adestrar ao dispositivo).

4. **Consistencia e estándares**. Séguense as normas e convencións xerais para o desenvolvemento de aplicacións Android:

   - Barra de acción na parte superior, con botón de retroceso na esquina esquerda e botón de preferencias na dereita.
   - Iconas estandarizadas.
   - Cadros de texto e etiquetas de tamaño abundante para ver nunha pantalla pequena.
   - Botóns grandes para facilitar a súa pulsación co dedo.

5. **Prevención de erros**. Un problema habitual é o uso indebido da aplicación por parte dos usuarios. A estratexia en FaceSampler consiste en tratar de reducir ao máximo a posibilidade de fallo cando o usuario proporciona datos ao sistema. Para elo, óptase por *inputs* de tipo *button*, *radio button*, *combo box* e *on/off button*. Este conxunto de entradas é dabondo para brindarlle ao usuario todas as funcionalidades previstas, incluíndo por suposto o control de preferencias.

6. **Minimización da carga da memoria do usuario**. Non existe ningunha acción que obrigue ao usuario a recordar ningún dato dalgún paso previo.

7. **Flexibilidade e eficiencia de uso**. A aplicación implementa procesos sinxelos para o usuario. Ademais, a navegación requirida é mínima, pois a interface componse de moi poucas pantallas.

8. **Diálogos estéticos e deseño minimalista**. Cada pantalla da aplicación amosa a información mínima imprescindible, evitando sobrecargar a interface. Nos diálogos, notificacións e demais alertas procúrase empregar sempre frases curtas e concisas.

9. **Axudar aos usuarios a recoñecer, diagnosticar e recuperarse de erros**. Android xestiona os erros de forma doada, notificando debidamente cada tipo de erro (ANRs, bloqueos, procesamento lento, etc.). A recuperación dos mesmos será inmediata, pois bastará con volver abrir a aplicación (moitas veces o propio sistema Android volverá a lanzala automaticamente).

10. **Axuda e documentación**. A aplicación caracterízase en gran medida pola facilidade de uso. Non obstante, está previsto elaborar un manual de usuario por se nalgún caso resulta necesario recorrer ao mesmo.

Erros identificados

A análise serviu para identificar algúns aspectos a mellorar no deseño da interface de usuario:

- Ao avaliar o cumprimento do principio 3 detectouse que poden darse ocasións nas que o usuario non teña a capacidade de reverter unha acción tomada por accidente. Concretamente, se se toca por erro o botón de iniciar adestramento, o sistema esquecerá no acto o adestramento previamente realizado, de existir.

  Solución: No caso de que se vaia a sobrescribir un adestramento previo, o sistema debe solicitar confirmación por parte do usuario cun cadro de diálogo. Esta nova característica engadiuse ao final da Pila de Produto e está actualmente á espera de ser implementada nun futuro *sprint*, xa fora do marco deste proxecto, posto que a súa prioridade é baixa.

- Ao avaliar o cumprimento do principio 9 recoñeceuse a carencia dunha personalización na xestión de erros. É certo que Android ofrece un sistema de notificación e reporte de incidencias moi bo, pero tal vez en ocasións sexa necesario amosar mensaxes de erro máis concretas, prescindindo das xenéricas que ofrece esta plataforma.

  Solución: Fixouse como necesidade identificar durante o desenvolvemento aqueles erros que precisasen dun trato personalizado e propio de FaceSampler. Na actualidade, rematado xa o desenvolvemento, identificouse tan só un: o bloqueo producido durante o arranque da aplicación por problemas de dependencias `[CU-1]`.

- Ao avaliar o cumprimento do principio 10 reafirmouse a necesidade de dispor dun manual de usuario.

  Solución: Redactouse un manual de usuario, que pode consultarse no Apéndice C.

5.3. Deseño da capa de control

A capa de Control responde a eventos e invoca peticións ao modelo cando se fai algunha solicitude sobre a información. Ditos eventos soen ser accións do usuario orixinadas ao interactuar coa capa de Presentación, pero tamén se xestionan aqueles eventos do sistema Android, como poden ser alarmas, rotación do dispositivo,

bloqueo do mesmo, etc. Esta capa da aplicación correspóndese co contido completo do paquete Java `com.citius.fernando.faceSampler.controller`[1].

Para unha boa comprensión do deseño deste controlador, semella oportuno expoñer unha breve xustificación a alto nivel de cada clase que o compón (polo menos das máis relevantes), de xeito previo á exposición de ningún tipo de diagrama estrutural. As tarxetas CRC empregadas durante o desenvolvemento poden ser o mellor xeito de logralo.

En primeiro lugar, para controlar as funcionalidades ofrecidas pola vista principal da aplicación emprégase a `MainActivity`, unha clase que estende da xa comentada `Activity` de Android (ver Sección 5.1) e que emprega dous fragmentos[2], un para cada modo de uso (adestramento e mostraxe):

CRC-1 `MainActivity`	
Autor:	Fernando Estévez
Responsabilidades:	• Controlar a interacción entre o usuario e a vista principal da aplicación (ver Figura 5.5).
Colaboradores:	• `TrainingFragment` • `SamplingFragment`

Para a xestión das preferencias, compre destacar as seguintes clases:

CRC-2 `SettingsActivity`	
Autor:	Fernando Estévez
Responsabilidades:	• Controlar a interacción entre o usuario e o panel de preferencias da aplicación (ver Figura 5.7). • Habilitar e deshabilitar a capacidade de manipular as preferencias en función das condicións dadas polo contexto e o dispositivo.
Colaboradores:	Non aplica.

CRC-3 `NumberPickerPreference`	
Autor:	Fernando Estévez
Responsabilidades:	• Controlar a interacción entre o usuario e as preferencias de tipo *number picker* (ver Figura 5.7b), non provistas por defecto no entorno Android.
Colaboradores:	Non aplica.

[1] Todo o código Java da aplicación localízase baixo a ruta `app/src/main/java` do proxecto Android Studio.

[2] Un fragmento (`android.app.Fragment`) representa un comportamento ou unha parte da interface de usuario nunha Activity. Pódense combinar varios fragmentos nunha soa actividade para crear unha IU multipanel e volver a usar un fragmento en múltiples actividades [31].

Finalmente, para o control dos procesos relativos á aprendizaxe, tanto no adestramento como na mostraxe, empréganse estas clases:

CRC-4 `ForegroundTrainingActivity`	
Autor:	Fernando Estévez
Responsabilidades:	• Xerar e actualizar de xeito dinámico a vista de adestramento en primeiro plano (ver Figura 5.6). • Controlar a interacción entre o usuario e dita vista. • Ordenar a captura de fotogramas e amosalos por pantalla en tempo real. • Solicitar información do entorno de forma periódica (estado do dispositivo). • Ordenar que se realice unha aprendizaxe en base aos estados e os fotogramas obtidos.
Colaboradores:	• `StateObtainerInterface [CRC-6]` • `ImageCaptorInterface [CRC-10]` • `StateAnalyzerInterface [CRC-16]`

CRC-5 `ForegroundSamplingService`	
Autor:	Fernando Estévez
Responsabilidades:	• Xerar e actualizar de xeito dinámico a notificación por pantalla (ver Figura 5.8) e demais sinais acústicos e visuais pertencentes ao proceso de mostraxe. • Controlar a interacción entre o usuario e a notificación ancorada á barra de notificacións. • Solicitar información do entorno de forma periódica (estado do dispositivo). • Ordenar que se tome unha decisión en canto a realizar unha nova captura ou non en base á información do banco de coñecemento e o estado do dispositivo en tempo de mostraxe.
Colaboradores:	• `StateObtainerInterface [CRC-6]` • `ImageCaptorInterface [CRC-10]` • `StateAnalyzerInterface [CRC-16]`

Desta última, `ForegroundSamplingService`, compre destacar que, ao contrario do resto de clases controladoras, non se trata dunha actividade, senón dun servizo[1], como ben indica o seu nome. Isto débese a que o proceso que controla non

[1]Un servizo (`android.app.Service`) é un compoñente dunha aplicación que pode realizar operacións de larga execución en segundo plano e que non proporciona unha interface de usuario. Outro compoñente da aplicación pode iniciar un servizo e continuará executándose en segundo plano aínda que o usuario cambie a outra aplicación [32].

depende directamente dunha vista, xa que se pode pechar a aplicación e este seguirá funcionando.

Comprendida a utilidade de cada unha das clases máis relevantes, podemos profundar un pouco máis no deseño. A Figura 5.10 amosa un diagrama de clases UML da totalidade da capa de Control. Non se está a omitir ningunha clase, aínda que só se amosan os atributos e métodos realmente significativos de cada unha delas para a súa comprensión co fin de non sobrecargar o diagrama. Cinco das clases reflectidas nel aparecen nun ton agrisado e non están contidas dentro de ningún dos paquetes da capa. Preténdese así indicar que son clases alleas á aplicación, provistas polo SDK Android e estendidas polas clases propias. Conviña incluílas na modelaxe para amosar as propiedades das mesmas que son empregadas polas clases propias.

No diagrama vese como a capa se subdivide en tres paquetes ben diferenciados. O paquete `main` contén aquelas clases empregadas no control da vista principal da aplicación. O paquete `preferences` inclúe aquelas outras destinadas ao control dos axustes. Finalmente, o paquete `learning` engloba a aquelas clases utilizadas para o control dos procesos relacionados coa aprendizaxe.

5.4. Deseño da lóxica de negocio

Esta capa é a parte do sistema que se encarga de codificar as regras de negocio do mundo real que determinan como a información pode ser creada, mostrada e cambiada. Como xa vimos na Sección 5.1 cando falabamos da arquitectura global, divídese en tres módulos diferenciados.

Nesta sección explicaranse de xeito individual os detalles de cada un deles. Para lograr a maior claridade posible, seguirase unha estratexia un tanto distinta á empregada na sección previa: darase unha xustificación da presencia das clases máis relevantes de cada módulo por medio de tarxetas CRC ou directamente cunha breve descrición, segundo conveña, pero primeiro presentaranse os respectivos diagramas de clases. Tamén se comentarán os patróns de deseño empregados. Recoméndase seguir a explicación textual tendo presente en todo momento o diagrama de clases correspondente.

5.4.1. Módulo de Recolección de datos

Este módulo subministra o estado do dispositivo `[RI-6]` baixo demanda. As peticións proveñen sempre da capa de Control da aplicación. A Figura 5.11 amosa o seu diagrama de clases ao completo, sen omitir ningunha clase. Por comodida-

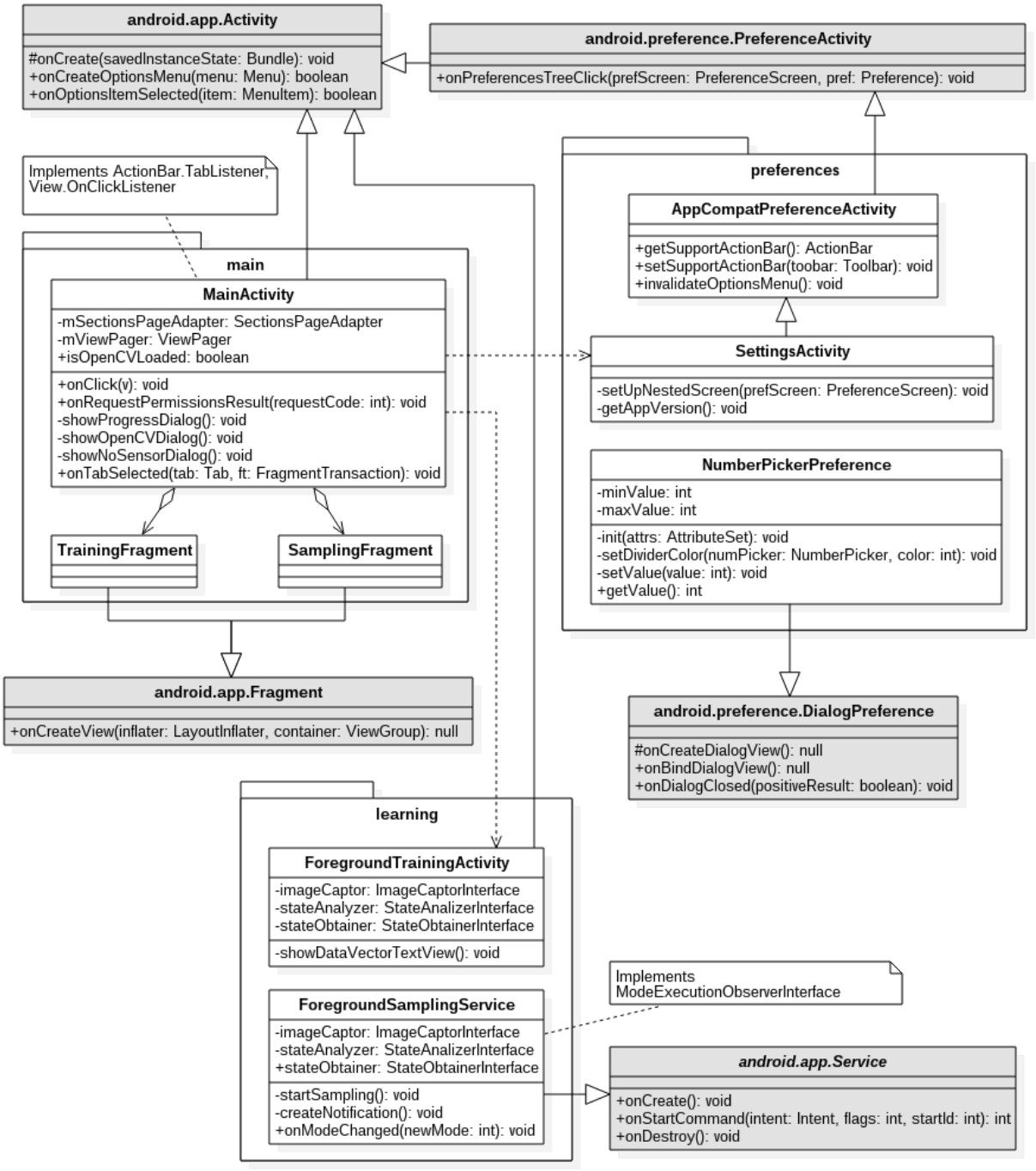


Figura 5.10: Diagrama de clases da capa de Control.

de de lectura e interpretación, inclúense só aquelas operacións e atributos máis relevantes.

O módulo proporciona ao exterior a interface `StateObtainerInterface`:

CRC-6 `StateObtainerInterface`	
Autor:	Fernando Estévez
Responsabilidades:	• Definir a "porta de entrada" ao módulo. • Proporcionar o estado do dispositivo `[RI-6]` en base á información do contexto baixo demanda.
Colaboradores:	Non aplica.

Internamente, a interface é implementada pola clase `StateObtainer`:

CRC-7 `StateObtainer`	
Autor:	Fernando Estévez
Responsabilidades:	• Dar soporte ás operacións definidas na interface `StateObtainerInterface`.
Colaboradores:	• `SensorDataCollector` • `ServiceDataCollector`

O estado é representado mediante o POJO[1] `State`. Disponse de dous paquetes illados para a recolleita da información. O paquete `sensorData` é o encargado de proporcionar os datos provintes dos sensores do móbil `[RI-3]`, mentres que `serviceData` prové a información dos servizos `[RI-4]`. As súas arquitecturas son moi simétricas: os dous seguen o **patrón de mensaxería publica-subscribe** [33]:

- En ambos casos se dispón dun POJO para manexar a información en cuestión: `SensorData` para os sensores e `ServiceData` para os servizos.
- Para obter a información empréganse as clases `SensorDataCollector` e `ServiceDataCollector`.
- Para recibir actualizacións en tempo real, o solicitante (`StateObtainer`) debe implementar unha interface: `SensorSubscriberInterface` no primeiro caso e `ServiceSubscriberInterface` no segundo.

Ao instanciar un `SensorDataCollector` proporciónase no construtor a propia clase instanciadora que implementa `SensorSubscriberInterface`. Cando se produce un cambio na información dos sensores, o `SensorDataCollector` efectúa unha chamada de *callback* ao seu invocador proporcionando un novo obxecto `SensorData`. No caso dos servizos o fluxo é o mesmo.

[1] Un POJO (*Plain Old Java Object*) é unha clase Java simple sen dependencias.

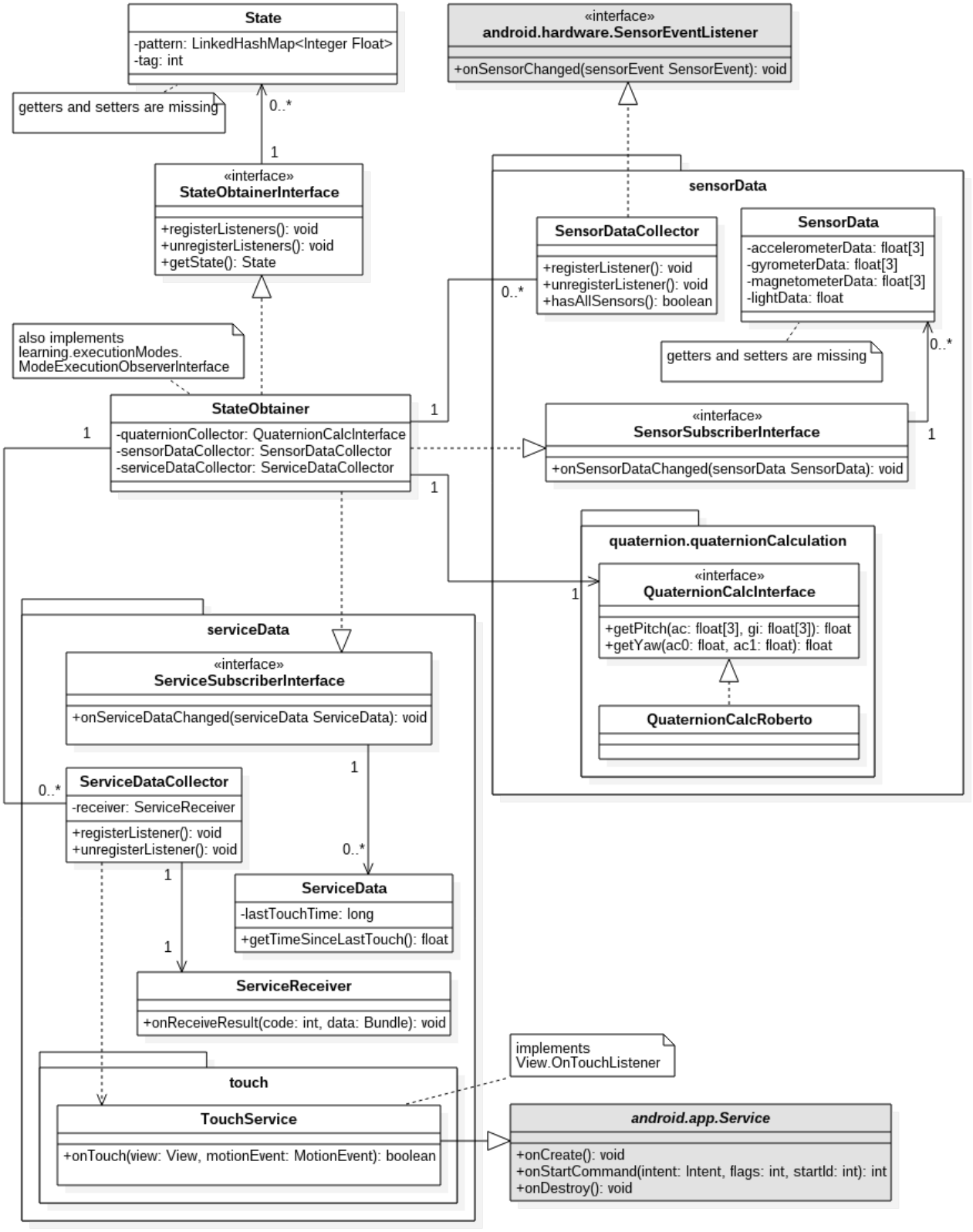


Figura 5.11: Diagrama de clases do módulo de Recolección de datos.

A maiores, cada un destes dous paquetes ten as súas peculiaridades. No primeiro, `sensorData`, atopámonos coa seguinte interface:

CRC-8 `QuaternionCalcInterface`	
Autor:	Fernando Estévez
Responsabilidades:	• Definir as operacións ás que se debe dar soporte para o cálculo do cuaternión e as proxeccións dos ángulos de rotación (*pitch* e *yaw*) empregados para representar a pose do dispositivo.
Colaboradores:	Non aplica.

En `serviceData` dispomos desta outra clase:

CRC-9 `TouchService`	
Autor:	Fernando Estévez
Responsabilidades:	• Rastrexar as pulsacións de pantalla, tanto cando a aplicación está en primeiro plano como cando non (excepto cando o dispositivo se bloquea), notificando cando se produce un novo toque.
Colaboradores:	Non aplica.

5.4.2. Módulo de Procesamento de imaxes

Este módulo componse á súa vez de dous sub-módulos diferenciados: un destinado á captura e almacenamento de imaxes (`capture`) e outro empregado para a detección de rostros (`detection`). Ámbolos dous empregan obxectos de tipo `ImageMatrix` para procesar, representar, transmitir e almacenar as imaxes. Na Figura 5.12 preséntase o diagrama de clases correspondente.

No sub-módulo de captura de imaxes recórrese de novo a unha arquitectura publica-subscribe:

CRC-10 `ImageCaptorInterface`	
Autor:	Fernando Estévez
Responsabilidades:	• Ser a porta de entrada ao sub-módulo e establecer as operacións que este ofrece ao exterior: tomar capturas `[RI-1]` baixo demanda.
Colaboradores:	Non aplica.

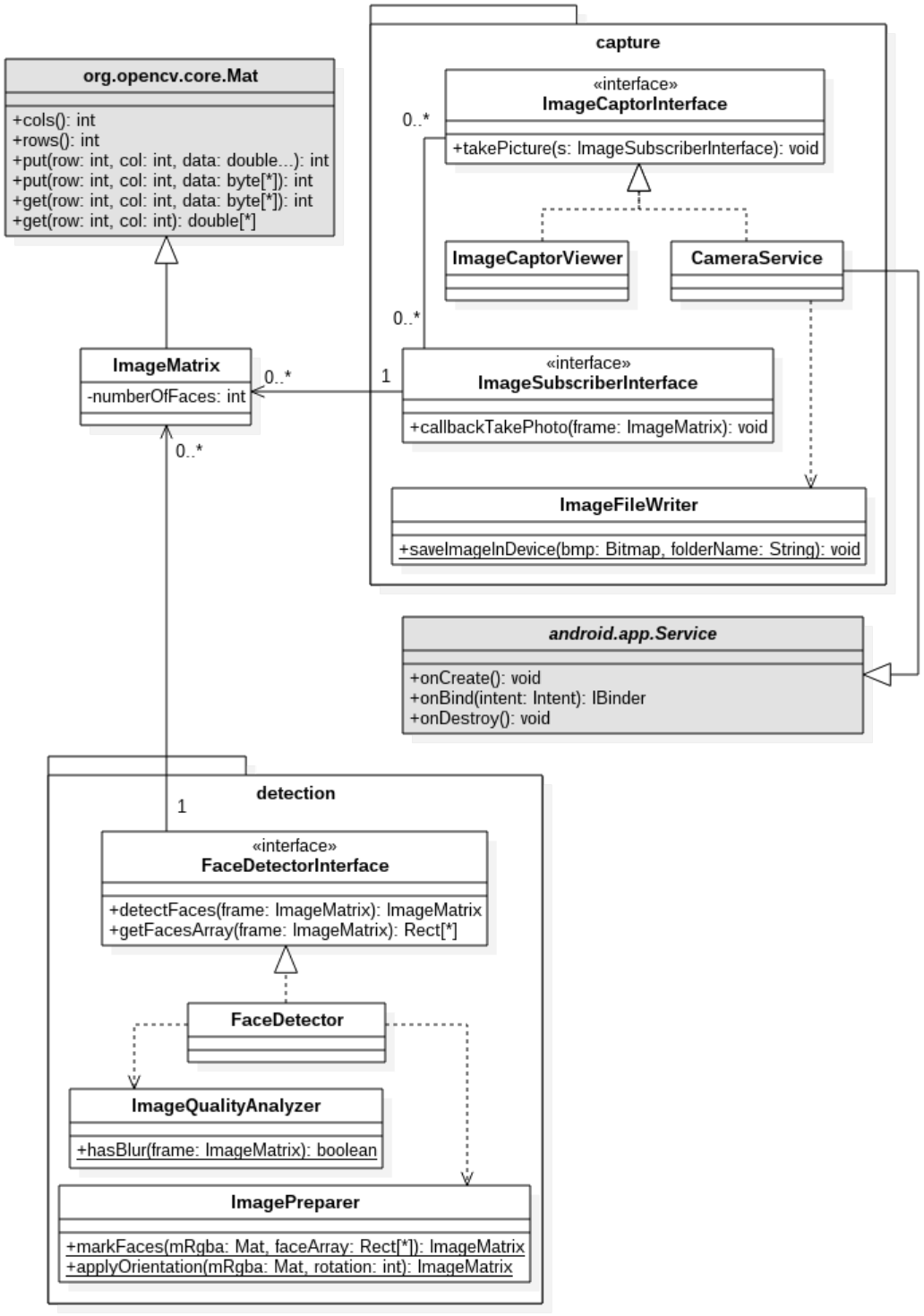


Figura 5.12: Diagrama de clases do módulo de Procesamento de imaxes.

CRC-11 ImageSubscriberInterface	
Autor:	Fernando Estévez
Responsabilidades:	• Definir os métodos que debe implementar todo aquel que desexe subscribirse ao sistema de captura de imaxes.
Colaboradores:	Non aplica.

A interface `ImageCaptorInterface` é implementada por dúas clases distintas, que empregan técnicas distintas de captación. A primeira delas é `ImageCaptorViewer`, pensada para a captura de xeito continuo, amosando os resultados por pantalla en tempo real. A segunda é `CameraService`, un servizo destinado a tomar imaxes puntuais incluso se a aplicación está en segundo plano ou o seu proceso principal (da interface gráfica) rematou. Este último emprega `ImageFileWriter` para gardar as capturas e mostras na memoria do dispositivo.

No sub-módulo de detección de rostros atopamos:

CRC-12 FaceDetectorInterface	
Autor:	Fernando Estévez
Responsabilidades:	• Servir como porta de entrada ao sub-módulo, para definir as operacións que este ofrece ao exterior: detectar rostros nas imaxes que o solicitante proporcione.
Colaboradores:	Non aplica.

CRC-13 FaceDetector	
Autor:	Fernando Estévez
Responsabilidades:	• Dar soporte ás operacións definidas por `FaceDetectorInterface`.
Colaboradores:	• `ImageQualityAnalyzer [CRC-14]` • `ImagePreparer [CRC-15]`

CRC-14 ImageQualityAnalyzer	
Autor:	Fernando Estévez
Responsabilidades:	• Identificar problemas relacionados coa calidade da imaxe. Ata a data, permítese detectar cando unha imaxe é borrosa (ten *blur*).
Colaboradores:	Non aplica.

CRC-15 ImagePreparer	
Autor:	Fernando Estévez

CRC-15 ImagePreparer *(cont.)*	
Responsabilidades:	• Marcar na imaxe as caras detectadas tal e como se describe no requisito non funcional [RNF-3]. • Rotar a imaxe e reverter a rotación. • Recortar a imaxe orixinal (captura) deixando só a cara do usuario (mostra).
Colaboradores:	Non aplica.

FaceDetectorInterface é implementada tan só por FaceDetector. Se nun futuro se desexa introducir un novo sistema de detección, este tamén deberá respectar a interface definida.

5.4.3. Módulo de Aprendizaxe

Este módulo contén a algoritmia necesaria para poder aconsellar á capa de Control na toma de decisións. A Figura 5.13 recolle o seu diagrama de clases UML. Nel podemos distinguir tres grandes paquetes: stateAnalyzer, algorithms e executionModes.

En stateAnalyzer sitúanse os analizadores de estados. A súa tarefa é estudar e contrastar o estado actual do dispositivo co *dataset* de adestramento baixo a supervisión do controlador con fins diversos. Debemos destacar as seguintes clases:

CRC-16 StateAnalyzerInterface	
Autor:	Fernando Estévez
Responsabilidades:	• Empregarse como porta de entrada ao módulo de Aprendizaxe. • Definir os métodos que todo analizador de estados debe soportar.
Colaboradores:	• StateObtainerInterface [CRC-6] • ImageCaptorInterface [CRC-10] • FaceDetectorInterface [CRC-12] • LearningAlgorithmFactory

CRC-17 TrainingStateAnalyzer	
Autor:	Fernando Estévez

CRC-17 `TrainingStateAnalyzer` *(cont.)*	
Responsabilidades:	• Dar soporte aos métodos definidos en `StateAnalyzerInterface`. • Crear o *dataset* de adestramento a partir da análise do estado do dispositivo en contraste coa información proporcionada polo módulo de Procesamento de imaxe.
Colaboradores:	Non aplica.

CRC-18 `SamplingStateAnalyzer`	
Autor:	Fernando Estévez
Responsabilidades:	• Dar soporte aos métodos definidos en `StateAnalyzerInterface`. • Predicir cando é un bo momento para tomar unha captura e obter unha mostra válida para o modelo de usuario contrastando o *dataset* de adestramento co estado actual do dispositivo.
Colaboradores:	Non aplica.

Os dous analizadores deseñados `[CRC-17, CRC-18]` ofrecen dúas funcionalidades totalmente distintas, aínda que ambas seguen o mesmo esquema. A súa inclusión supón un alivio na carga lóxica da capa de Control, podendo considerarse *helpers* do mesmo (patrón ***delegation***). Os analizadores actuarán periodicamente de xeito independente ao controlador, e só se deterán baixo a súa orde explícita.

O paquete `algorithms` proporciona os algoritmos de toma de decisións implementados. Para elo integra dous coñecidos patróns de deseño: ***singleton*** e ***factory***. A `LearningAlgorithmFactory` é a factoría que proporciona o algoritmo axeitado en cada momento, atendendo ao contexto. Todo algoritmo que se desexe integrar na aplicación debe implementar a interface `LearningAlgorithmInterface`. Desde o panel de preferencias, o usuario poderá escoller o algoritmo que desexe empregar e, ao momento, a factoría comezará a subministrar a instancia correspondente. Para evitar instanciacións innecesarias, tanto `LearningAlgorithmFactory` como cada un dos algoritmos empregan o patrón *singleton* para proporcionar un único punto de acceso global a cada elemento. Este deseño facilita a escalabilidade algorítmica solicitada `[RNF-13]`.

Para rematar, o paquete `executionModes` é o encargado de xestionar e notificar os cambios de modo entre execución normal e aforro de enerxía que se efectúan no proceso de mostraxe. Introdúcese aquí o patrón ***observer***, onde `ExecutionModeHandler` é a clase observada, que controla os cambios efectuados, mentres que os observadores serán todas aquelas clases que implementen a interface `ModeExecutionObserverInterface`. Actualmente, os observadores do

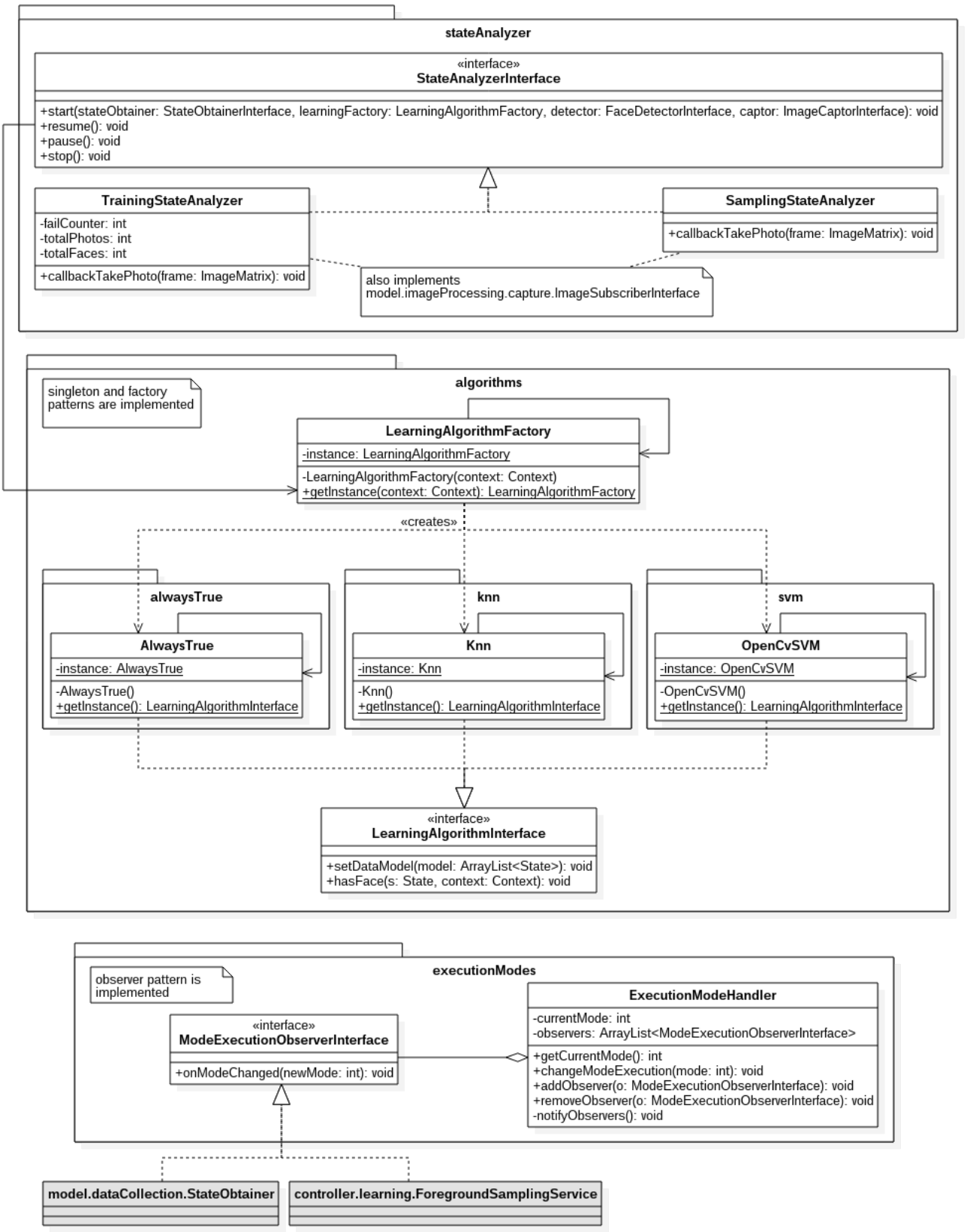


Figura 5.13: Diagrama de clases do módulo de Aprendizaxe.

modo de execución son dous: `StateObtainer` do módulo de recolección de datos e `ForegroundSamplingService` da capa de Control.

5.5. Descrición da interacción

Para concluír coa documentación do deseño, nesta sección achégase unha descrición da interacción entre os distintos compoñentes presentados ao longo do capítulo. Para elo, recórrese ao emprego de diagramas de secuencia UML. Dada a inviabilidade dunha representación exhaustiva de dita interacción (cumpriría, cando menos, un diagrama de secuencia por cada caso de uso descrito na Sección 2.4), decidiuse plasmar o fluxo orixinado durante os dous procesos principais (e tamén máis complexos): adestramento e mostraxe.

Na Figura 5.14 amósase o diagrama de secuencia do proceso de adestramento en primeiro plano. Abarca os tres casos de uso relacionados co mesmo `[CU-10, CU-11, CU-12]`. O adestramento comeza coa solicitude do usuario a través da interface gráfica. Na capa de control, `MainActivity` xestiona a petición e solicita a creación dunha `ForegroundTrainingActivity` a través dun *intent*[1] de Android. Se o dispositivo conta cunha versión Android 6.0 (Marshmallow) ou superior, os permisos de acceso á cámara e ao almacenamento interno terán que pedirse de xeito dinámico[2]. `ForegroundTrainingActivity` instancia as dependencias necesarias e comeza o proceso de adestramento. Periodicamente, o analizador realiza a chamada asíncrona `10: takePicture(this)` ao sistema de captura. Na resposta, obtense o estado actual, devólvese a imaxe e compróbase que esta é o suficientemente nítida (chamada estática `14: hasBlur(frame)`). Só en caso afirmativo se procede a detectar os rostros na mesma. Aínda que o diagrama non o recolle explicitamente, chegados a este punto etiquétase o estado como un éxito ou un fracaso en base ás condicións dadas. Finalmente, comunícase a imaxe procesada á vista. Este proceso repítese durante todo o tempo que o usuario desexe, ata que prema o botón de retroceso, finalizando así o adestramento. O diagrama da Figura 5.14 non recolle outras accións como as contempladas nos casos de uso de pausar a aplicación `[CU-5]` ou retomala `[CU-6]` porque a complexidade do mesmo aumentaría notoriamente e non cumpriría co seu obxectivo, que é o de axudar na comprensión da interacción orixinada neste escenario concreto.

[1] Un *intent* (`android.content.Intent`) é un obxecto de acción que se pode empregar para solicitar unha acción de outro compoñente da aplicación. Aínda que os *intents* facilitan a comunicación entre os compoñentes de moitas maneiras, existen tres casos de uso fundamentais: comezar unha actividade, comezar un servizo e entregar unha mensaxe [34].

[2] Ditos permisos solicitaranse polo menos a primeira vez que se realice o adestramento ou a mostraxe. Se o usuario marca a opción de "lembrar elección" non se lle volverán a solicitar. Decidiuse plasmar esta comprobación de permisos no diagrama xa que engade valor ao mesmo sen introducir demasiada complexidade.

A Figura 5.15 presenta o diagrama de secuencia do proceso de mostraxe, abarcando os tres casos de uso relacionados co mesmo `[CU-7, CU-8, CU-9]`. O comezo é moi semellante ao do caso anterior: o usuario solicita que se inicie a mostraxe e a `MainActivity` comproba os permisos de ser necesario. Neste caso, solicítase crear un `ForegroundSamplingService`, tamén mediante un *intent*. O proceso de análise desta vez é diferente: comézase obtendo o estado para logo facer a predición. As chamadas á `LearningAlgorithmFactory` e ao algoritmo en cuestión son estáticas sempre. No diagrama inclúese a interface `LearningAlgorithmInterface` xa que o algoritmo a empregar dependerá das condicións do contexto. Logo da chamada `20: takePicture(this)` omítese o *callback* que realiza o sistema de captura, xa que engade demasiada complexidade ao diagrama. Nese *callback* comprobaríase se realmente a imaxe é un éxito, de igual xeito a como se fai na Figura 5.14. Omítense tamén a almacenaxe das capturas e mostras en memoria. Igual que no caso do adestramento, non se contemplan outras casuísticas relacionadas (permutacións entre modos de execución) porque é inviable.

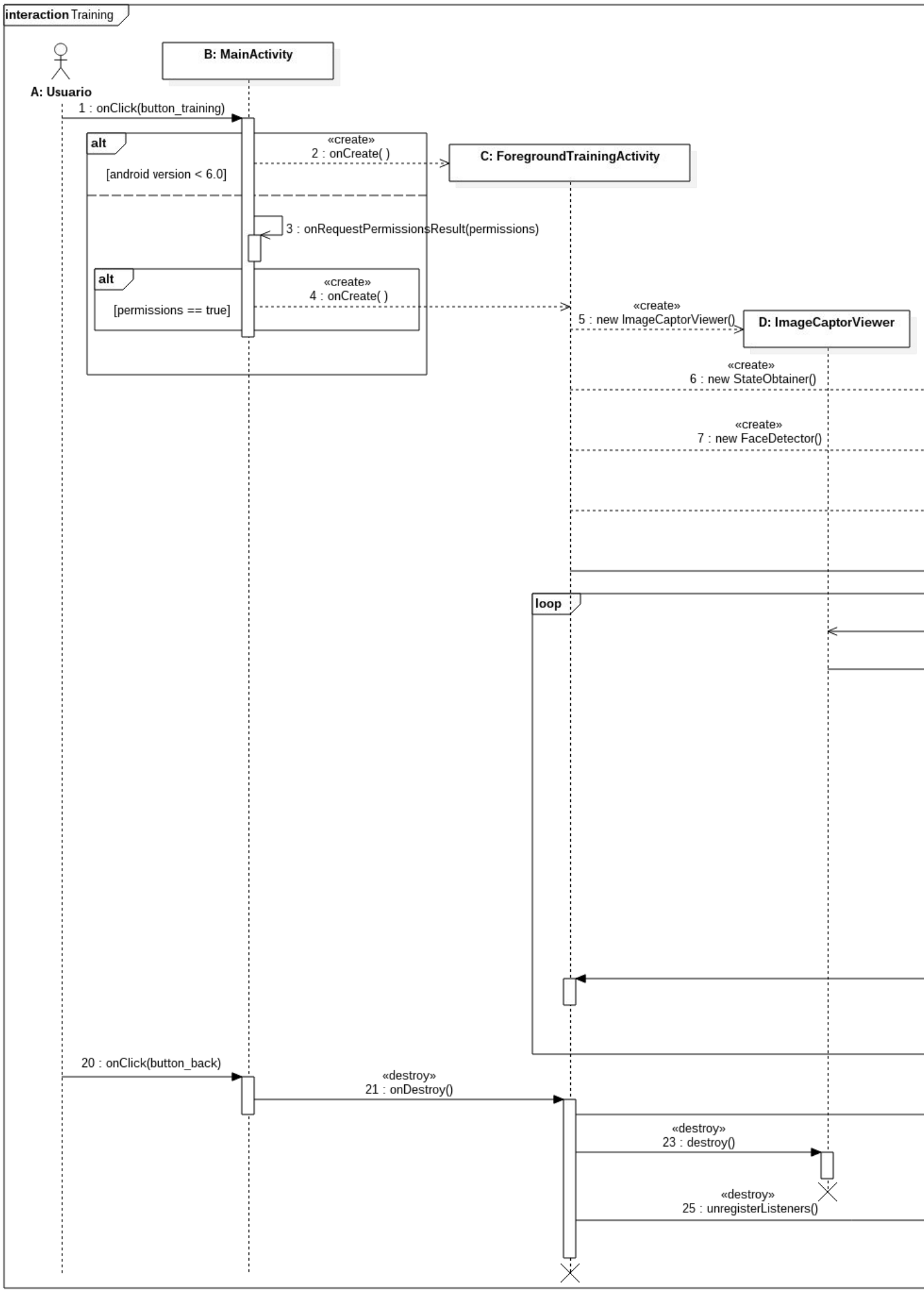
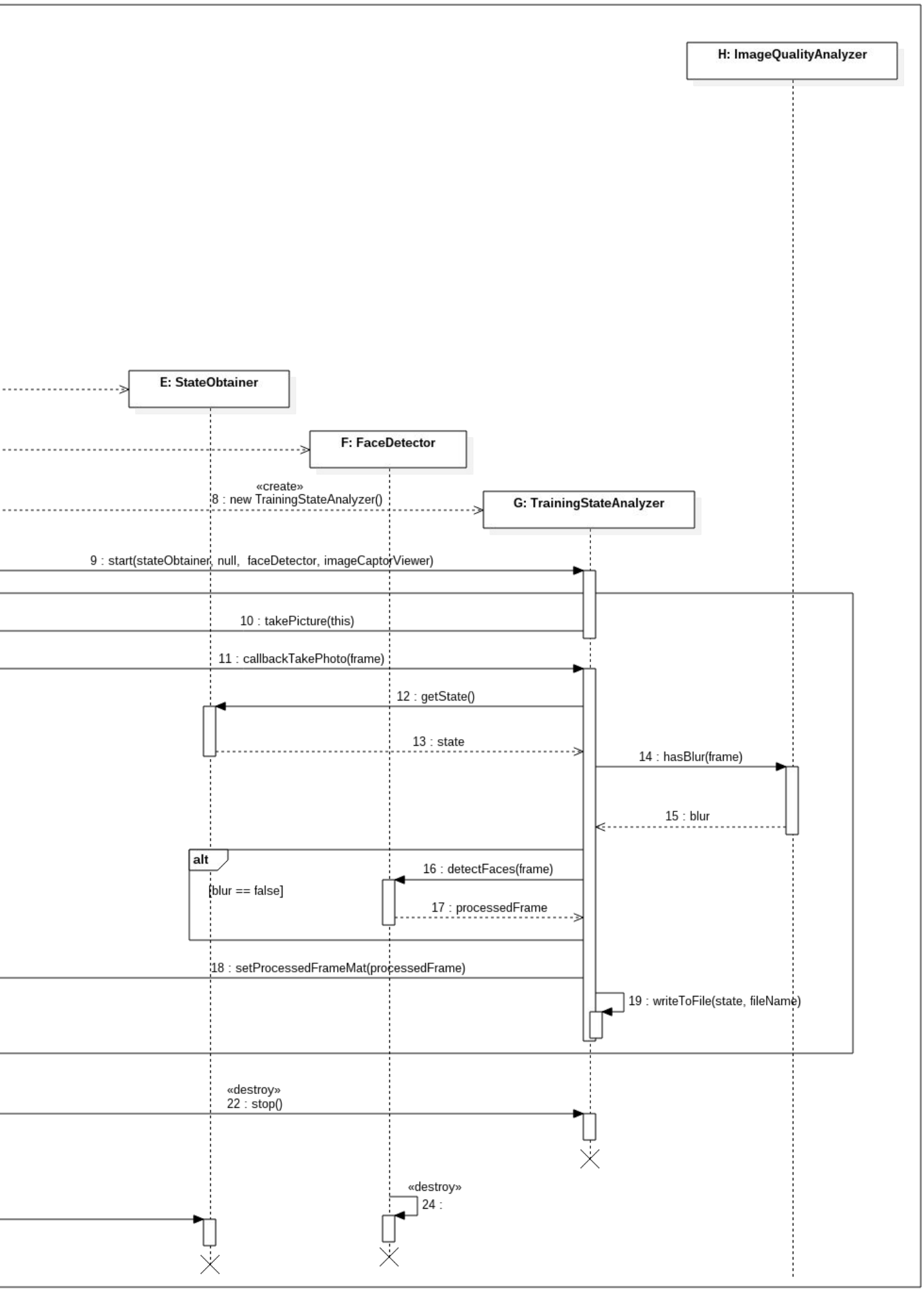


Figura 5.14: Diagrama de secuencia do proceso de adestramento.


H: ImageQualityAnalyzer
E: StateObtainer
F: FaceDetector
«create»
8 : new TrainingStateAnalyzer()
G: TrainingStateAnalyzer
9 : start(stateObtainer, null, faceDetector, imageCaptorViewer)
10 : takePicture(this)
11 : callbackTakePhoto(frame)
12 : getState()
13 : state
14 : hasBlur(frame)
15 : blur
alt
[blur == false]
16 : detectFaces(frame)
17 : processedFrame
18 : setProcessedFrameMat(processedFrame)
19 : writeToFile(state, fileName)
«destroy»
22 : stop()
«destroy»
24 :

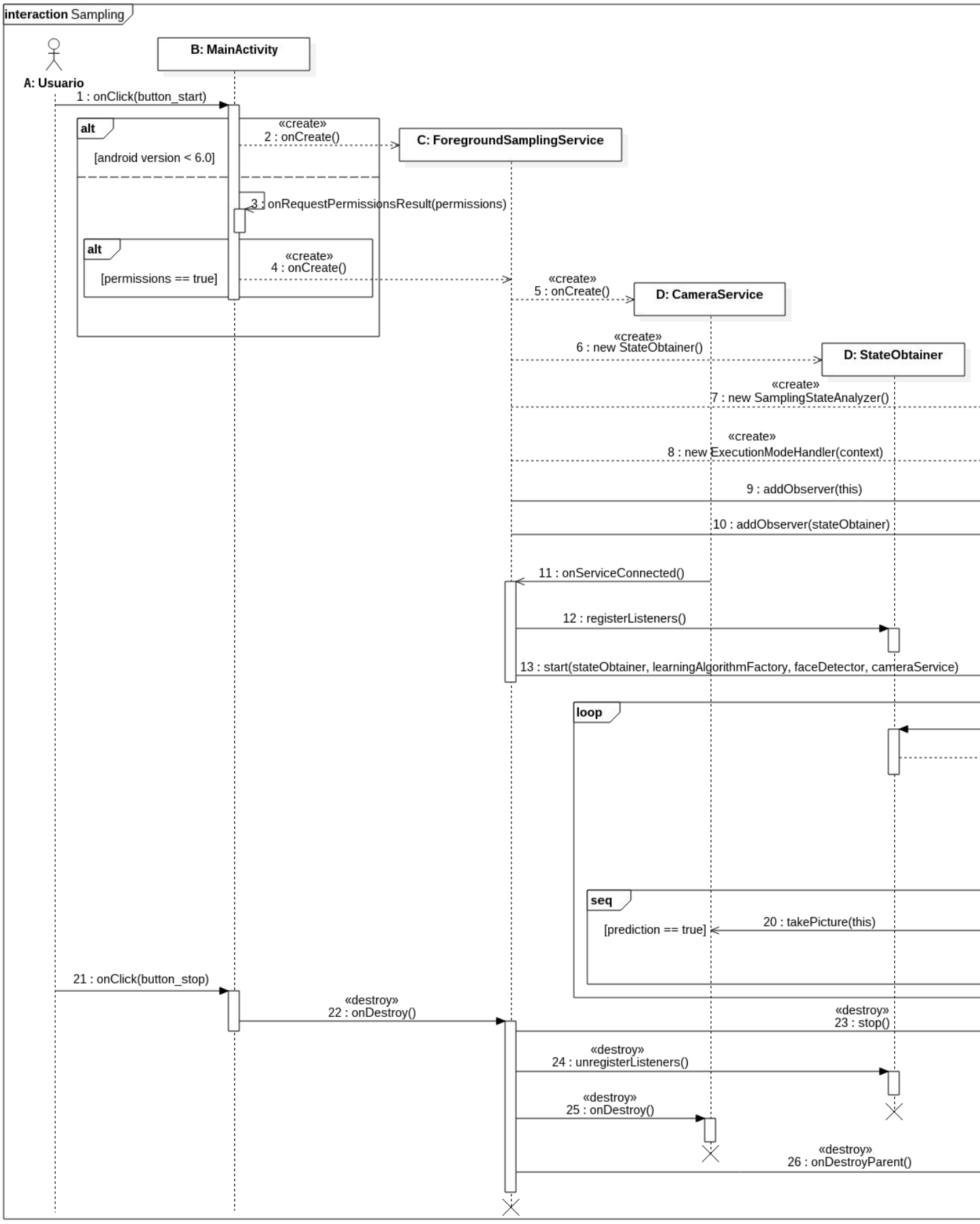
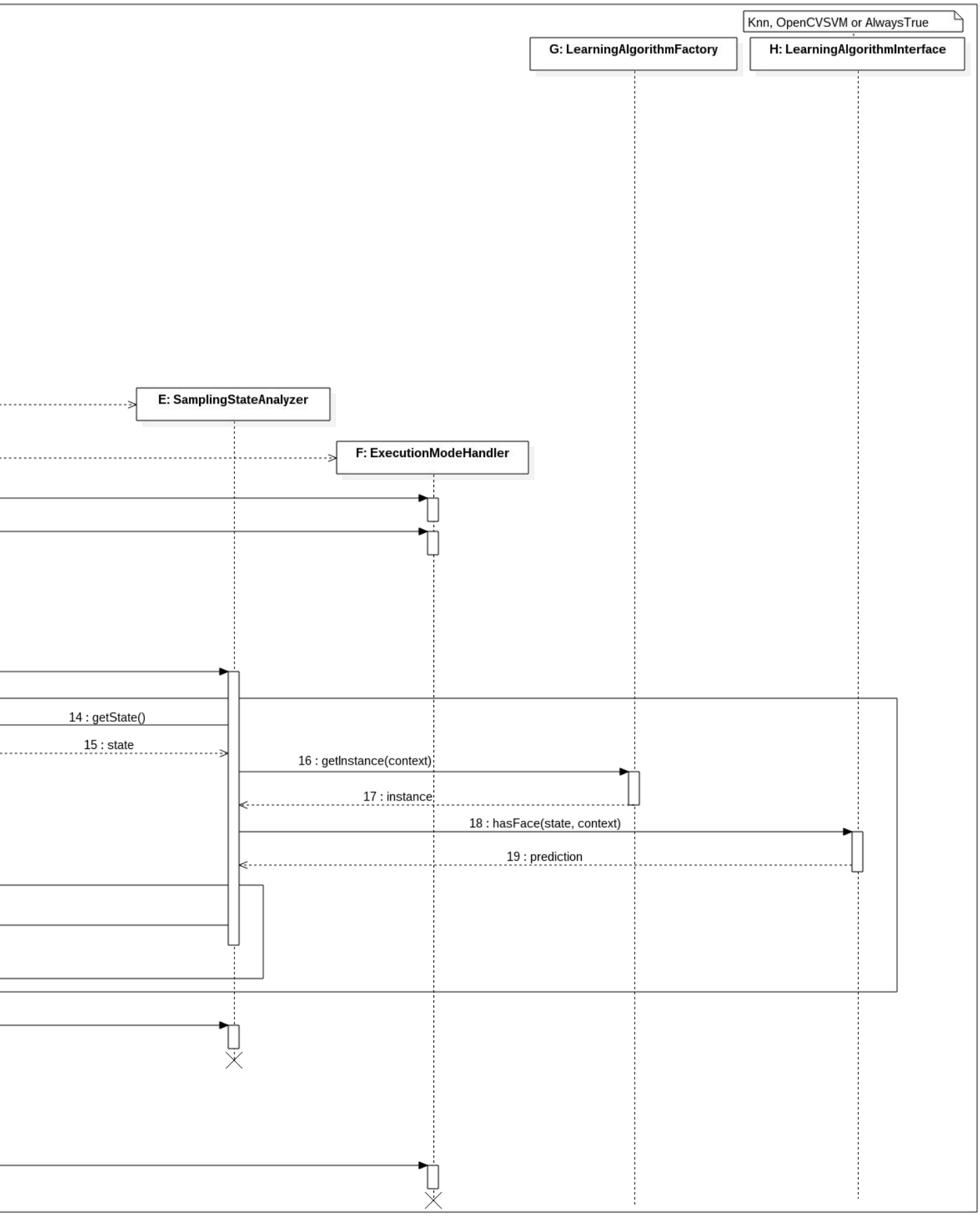


Figura 5.15: Diagrama de secuencia do proceso de mostraxe.


Knn, OpenCVSVM or AlwaysTrue
G: LearningAlgorithmFactory
H: LearningAlgorithmInterface
E: SamplingStateAnalyzer
F: ExecutionModeHandler
14 : getState()
15 : state
16 : getInstance(context)
17 : instance
18 : hasFace(state, context)
19 : prediction

Capítulo 6

Execución de probas e beta privada

Neste capítulo preséntase o resultado da execución do plan de probas definido no Capítulo 3. En primeiro lugar, na Sección 6.1, exporase o informe de execución das probas. Posteriormente, na Sección 6.2, comentaranse os resultados obtidos coa beta privada. Por último, na Sección 6.3, presentarase un informe coas incidencias detectadas.

6.1. Informe de execución das probas

A execución das probas funcionais ao longo dos sucesivos *sprints* permitiu a identificación e a corrección de cantidade de erros. Deste xeito, antes do peche de cada *sprint* e da correspondente revisión co cliente garantiuse o cumprimento da cobertura mínima estipulada no plan de probas. Non se aporta documentación da execución destas probas, pois compre lembrar que se realizan de xeito paralelo ao desenvolvemento, antes de fixar a liña base da versión do produto resultado do *sprint* en cuestión. Os cambios que xorden da detección de fallos con este tipo de probas son sempre cambios menores, corrixidos no momento, pois é o único xeito de poder superar o *sprint* correspondente.

A Táboa 6.1 resume a execución das probas estruturais, realizadas en Android Studio con JUnit e Mockito. Superáronse 16 casos de proba, dun total de 19. As incidencias asociadas a cada un dos 3 casos fallidos recóllense cara o final deste capítulo, na Sección 6.3.

As probas aleatorias executáronse sen maior complicación en grupos de catro desde a consola de Firebase Test Lab. Finalizada a proba, Firebase ofrece un informe de execución máis que elaborado. En primeiro lugar, permite ver os

Proba	Caso de proba		Duración	Resultado	Incidencias
PE-1	CP-1	testStartService	0 ms	Ok	
	CP-2	testWithBoundService	1280 ms	Ok	
PE-2	CP-3	testKNNHasFaceFalseSimple	0 ms	Ok	
	CP-4	testKNNHasFaceTrueSimple	0 ms	Ok	
	CP-5	testKNNHasFaceFalse	0 ms	Ok	
	CP-6	testKNNHasFaceTrue	0 ms	Ok	
	CP-7	testSVMHasFaceFalseSimple	0 ms	Ok	
	CP-8	testSVMHasFaceTrueSimple	0 ms	Ok	
	CP-9	testSVMHasFaceFalse	0 ms	Erro	INC-1
	CP-10	testSVMHasFaceTrue	0 ms	Ok	
PE-3	CP-11	testNothingHappens	0 ms	Ok	
	CP-12	testUserWantsLEMode	0 ms	Ok	
	CP-13	testUserWantsNormalMode	0 ms	Ok	
	CP-14	testUserWantsLEModeAndNormalModeAgain	1 ms	Ok	
	CP-15	testUserWantsNormaModeAndLEModeThen	50 ms	Ok	
	CP-16	testUserPresent	51 ms	Erro	INC-2
	CP-17	testScreenOff	1 ms	Erro	INC-2
	CP-18	testStopFromNormal	1080 ms	Ok	
	CP-19	testStopFromLE	25 ms	Ok	

Táboa 6.1: Resumo de execución das probas estruturais.

rexistros (*logs*) efectuados durante o proceso. De producirse un fallo, amosa a traza completa do mesmo como se estivésemos a executar a aplicación desde o IDE, tal e como se pode ver na Figura 6.1.

Ademais, esta ferramenta filma o proceso de execución, para posteriormente proporcionar capturas de pantalla do mesmo, así coma un vídeo completo da proba. Neste último caso, o contido multimedia é complementado con gráficos de rendemento (CPU, memoria e rede), como se pode ver na Figura 6.2. Finalmente, se se desexa, tamén se pode consultar o mapa de actividade que resume a execución en forma de grafo. A Figura 6.3 amosa dous exemplos destes mapas de actividade. Dado o seu gran tamaño amósanse en miniatura, pudendo consultalos a escala real en formato dixital (ver Apéndice B).

A Táboa 6.2 resume os resultados obtidos. Superáronse 14 das 16 probas definidas. De novo, as incidencias asociadas a cada fallo poden ser consultadas na Sección 6.3.

6.2. Beta privada

Tal e como se expuña no plan de probas (ver Sección 3.6.1), púxose en funcionamento unha versión beta privada da aplicación co fin de medir a compatibilidade da mesma sobre distintos dispositivos e versións de Android, así como detectar aquelas incidencias que puidesen manifestarse. A explotación desta beta efectuouse no período do 08/05/2017 ao 12/06/2017. Durante este tempo, unha serie

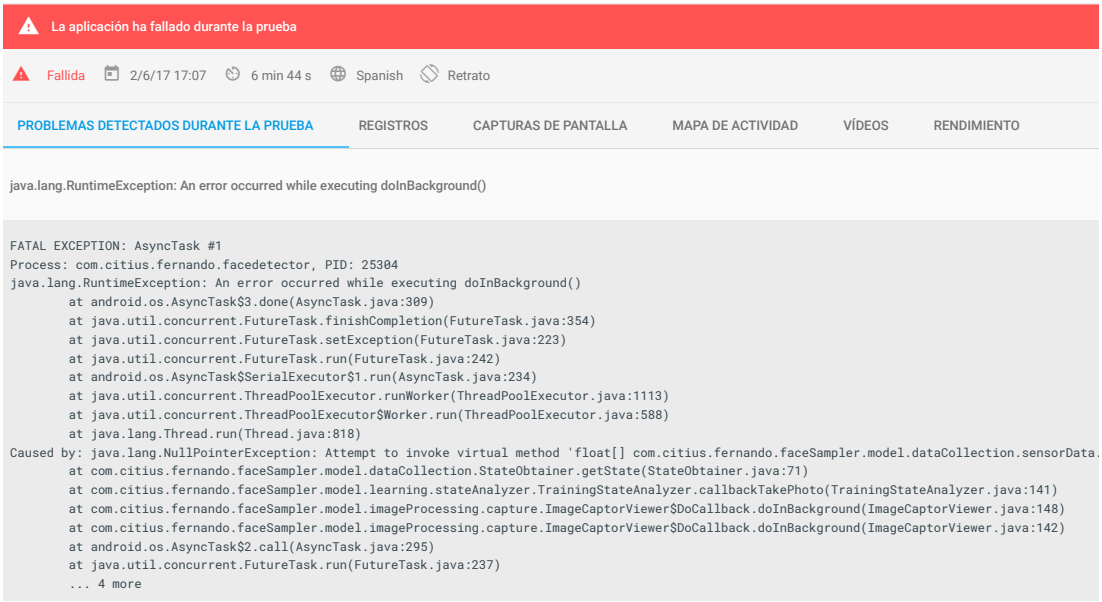


Figura 6.1: Exemplo de traza dun fallo reportado por Firebase.

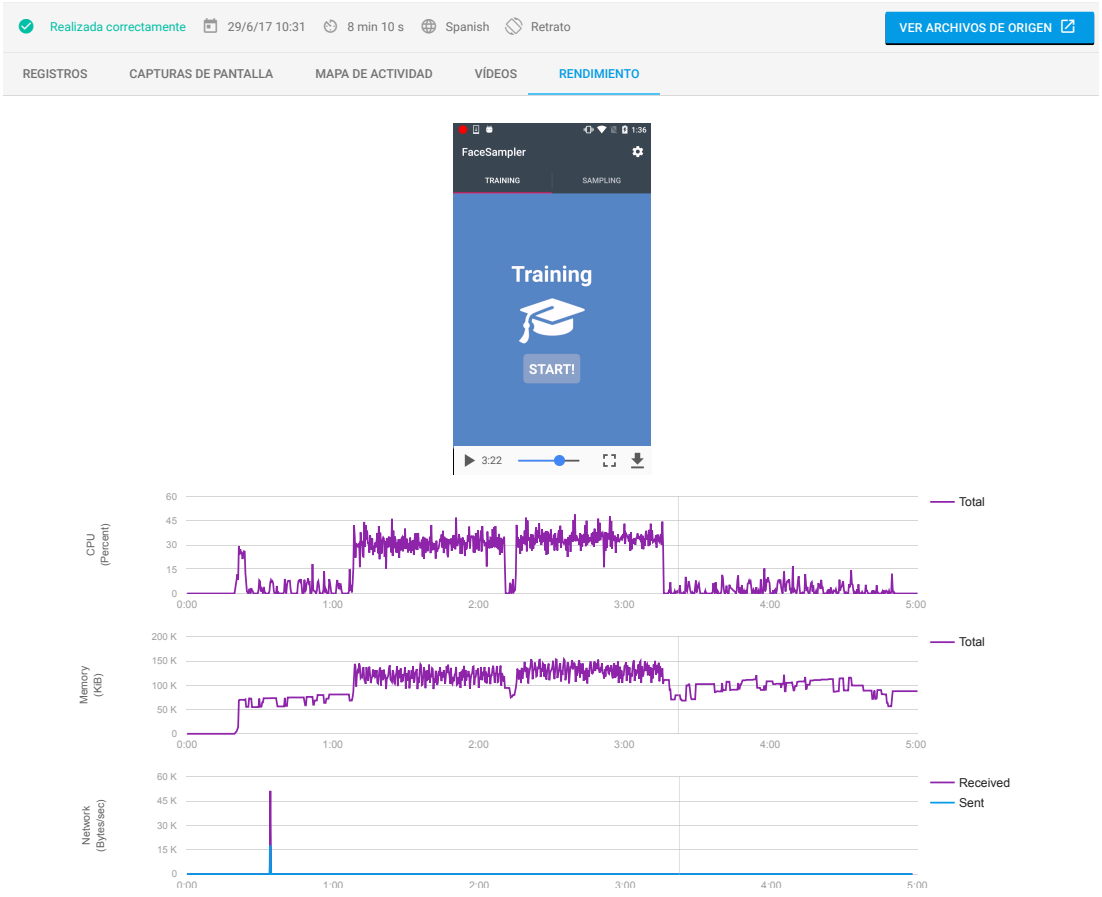
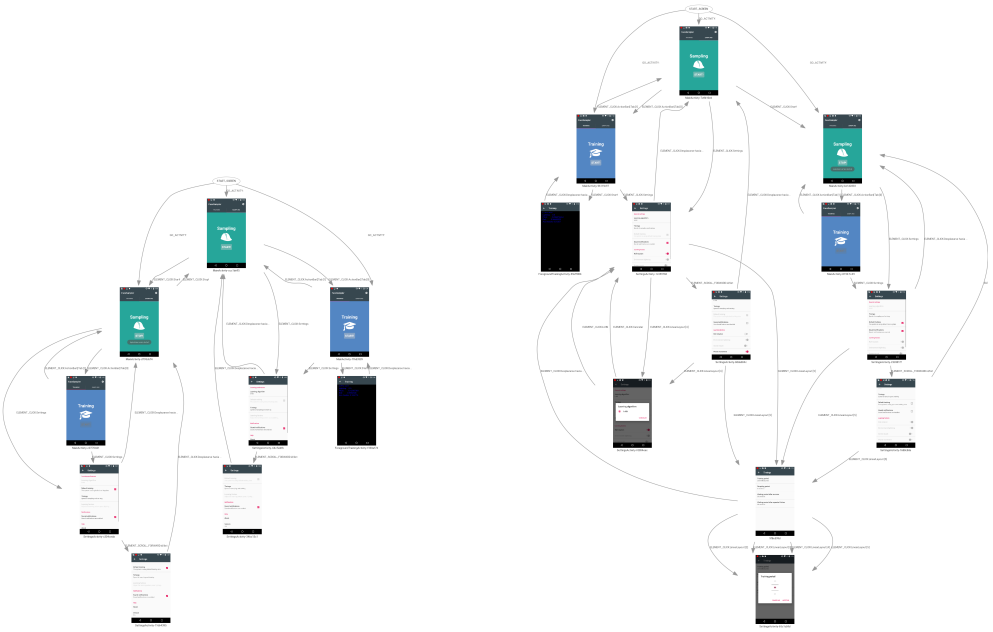


Figura 6.2: Exemplo de análise de rendemento en Firebase.



(a) Execución de `PA-6`. **(b)** Execución de `PA-10`.

Figura 6.3: Exemplos de mapas de actividade de Firebase (miniaturas).

Proba	Dispositivo	v. Android	Orientación	Duración	Resultado	Incidencias
PA-1	Galaxy Note 3	4.4	Retrato	38 s	Erro	INC-3
PA-2	Google Nexus 7	5.0	Retrato	5 min 21 s	Ok	
PA-3	Google Nexus 9	5.0	Retrato	4 min 11s	Ok	
PA-4	Sony Xperia Z3	5.0	Retrato	4 min 51s	Ok	
PA-5	OnePlus One	5.1	Retrato	4 min 49 s	Ok	
PA-6	Google Nexus 6	5.1	Retrato	4 min 15 s	Ok	
PA-7	Galaxy Note 4	5.1	Retrato	4 min 15 s	Ok	
PA-8	Motorola Moto G4	6.0	Retrato	2 min 18 s	Ok	
PA-9	Galaxy S7 edge	6.0	Retrato	3 min 29 s	Erro	INC-4
PA-10	Google Nexus 5	6.0	Retrato	5 min 20 s	Ok	
PA-11	Google Nexus 5	6.0	Paisaxe	4 min 50 s	Ok	
PA-12	Sony Xperia X	6.0	Retrato	4 min 39s	Ok	
PA-13	Galaxy S7	7.0	Retrato	3 min 26 s	Ok	
PA-14	Google Pixel	7.1	Retrato	5 min 15 s	Ok	
PA-15	Google Nexus 5X	7.1	Retrato	5 min 9 s	Ok	
PA-16	Google Nexus 5X	7.1	Paisaxe	5 min 8 s	Ok	

Táboa 6.2: Resumo de execución das probas aleatorias.

de voluntarios tiveron acceso ás funcionalidades de FaceSampler. Nesta sección comentarase como foi o seu proceso de posta a punto, así como os resultados obtidos.

6.2.1. Posta a punto

En primeiro lugar, era necesario dispor dun método de distribución do instalador da ferramenta. Decidiuse empregar a plataforma **Google Play Store** por diversas razóns. Por un lado, é o xeito máis habitual e seguro polo cal os usuarios de dispositivos Android poden descargar as aplicacións que desexan; trátase, porén, dunha tenda *online* coñecida de xeito obrigado por todos os usuarios deste ecosistema. Polo outro, o feito de publicala pola vía de Google permite empregar a **Google Play Console** para administrar todas as fases da publicación, facer un seguimento dos usuarios, xestionar as versións, controlar a distribución e recompilar estatísticas antes e despois do lanzamento. A Figura 6.4 amosa un fragmento do panel principal de control desta consola, que recolle datos estatísticos sobre instalacións, usuarios, valoracións e bloqueos.

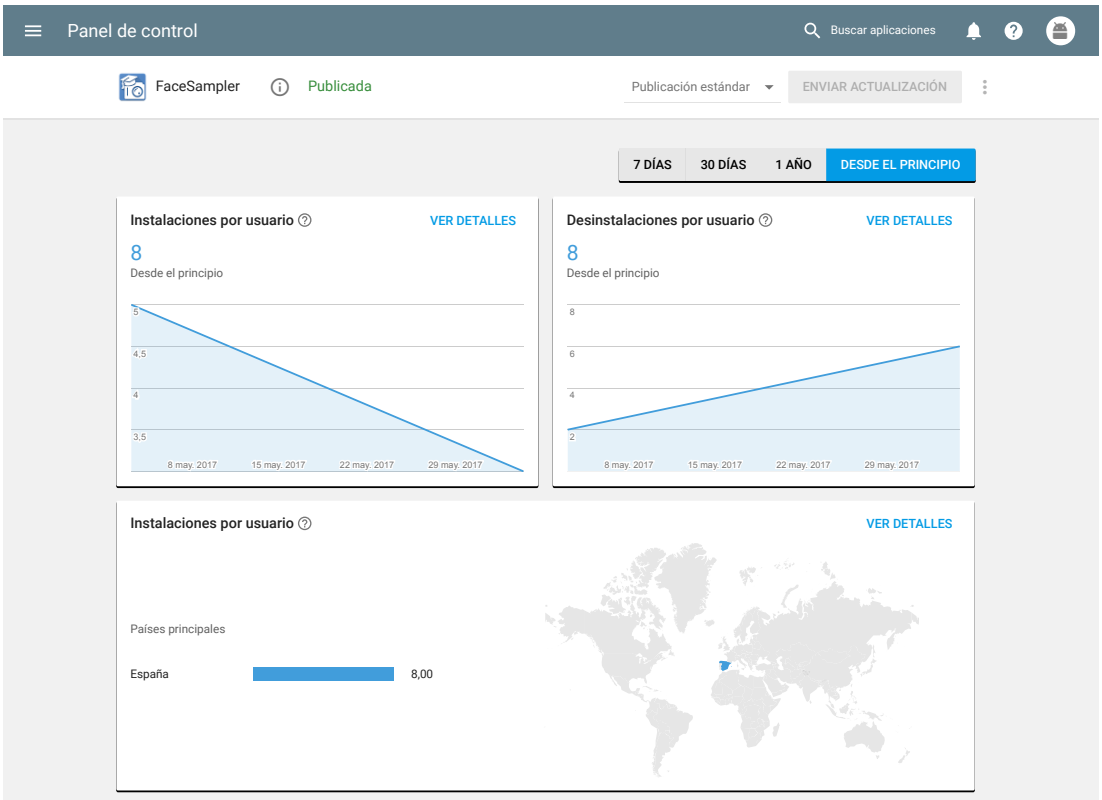


Figura 6.4: Panel de control de Google Play Console.

Desexábase garantir que todos os usuarios seleccionados fixesen un correcto uso da ferramenta. Sen embargo, non todos eles ían estar necesariamente familiarizados co recoñecemento de patróns. O feito de obrigar aos voluntarios a adestraren os seus dispositivos podía provocar que os adestramentos fosen pouco exhaustivos e, en consecuencia, que a mostraxe fose deficiente. Neste punto foi cando se introduciu o caso de uso "Xestionar o adestramento por defecto" `[CU-15]`, incluíndo no instalador da ferramenta un *dataset* de adestramento "de fábrica", xerado co dispositivo de desenvolvemento (o Bq Aquaris E5s). Deste xeito, unha vez instalada a aplicación no dispositivo dos usuarios podíase comezar inmediatamente a mostraxe, sen necesidade dun adestramento previo. Evidentemente, se o usuario o desexase, sempre podería realizar el o adestramento e seleccionar que *dataset* empregar desde o panel de preferencias, tal e como se describe no caso de uso antes citado.

De novo para garantir a eficiencia da mostraxe, algúns axustes foron bloqueados para forzar aos usuarios a empregar a mesma configuración na aprendizaxe. Estableceuse k-NN como algoritmo a utilizar por ser prudente na toma de decisións. Ademais, o conxunto de datos de entrada foi configurado para incluír exclusivamente aquelas variables que realmente aportasen valor na aprendizaxe. Tan só quedou excluído o tempo transcorrido desde o último toque de pantalla. O motivo da exclusión foi que para poder instruír ao dispositivo en base a toques de pantalla de xeito natural necesitaríase levar a cabo un adestramento en segundo plano durante un tempo prolongado, realizando tarefas habituais e cotiás; sen embargo, o noso sistema está pensado para aplicar adestramentos rápidos e en primeiro plano, de cara a demostracións, que non permiten simular unha interacción normal co dispositivo no que respecta ao toque de pantalla. Para que todos os usuarios empregasen esta configuración e non fose modificada por ninguén, deshabilitouse a sección de preferencias relativa ás entradas da aprendizaxe `[CU-16]`.

6.2.2. Resultados obtidos

Durante este período, un total de **8 voluntarios** usaron a aplicación de forma continua no seu día a día. Pedíuselles a todos agardar ata que a ferramenta obtivese un banco de **220 mostras** (*samples* `[RI-2]`) das súas caras. O tempo de emprego por usuario variou en función de diversos factores, como a frecuencia de uso do dispositivo, as condicións de contexto ás que se someteu a aplicación para tomar as fotos, á efectividade do adestramento por defecto en cada dispositivo, etc. Hai individuos que tan só precisaron tres días para acadar ese número de mostras, mentres que outros requiriron de tres semanas. A Táboa 6.3 amosa a variedade de dispositivos nos que se probou a aplicación, así como a versión Android de cada un.

Dispositivo	Versión Android
Samsung Galaxy Note 3	5.0
LG G4	5.1
Bq Aquaris E5s	5.1
OnePlus One	6.0
Asus Zenfone Selfie	6.0
HTC One M9	6.0
Nexus 5X	7.1
Bq Aquaris X5 Plus	7.1

Táboa 6.3: Dispositivos da beta privada.

De media, a taxa de acertos foi do **45,32 %**. Isto implica que, practicamente, por cada dúas capturas realizadas, unha serviu como mostra da cara do usuario. É dicir, que o sistema necesita tomar en torno a 490 capturas para poder quedarse con 220 mostras. Agora ben, compre destacar que a configuración empregada nesta beta volveu ao sistema **moi conservador**, de xeito que practicamente a totalidade das imaxes escollidas como mostras son perfectamente válidas e de calidade, pero moitas veces descartáronse outras que, de rebaixar un pouco os requisitos, serían tamén aceptadas sen problema. Vexamos algúns exemplos.

Na Figura 6.5 amósanse cinco mostras do modelo obtido para un único usuario. Pódese apreciar que a calidade de todas elas cumpre con creces os mínimos desexados: conteñen o rostro do usuario perfectamente encadrado, coa iluminación e nitidez suficientes. Evidentemente, algunhas son máis nítidas e outras máis borrosas, así como unhas son máis claras e outras máis escuras, pero esta variedade, orixinada pola dependencia co contexto, é precisamente do que desexamos dispor nos nosos modelo.

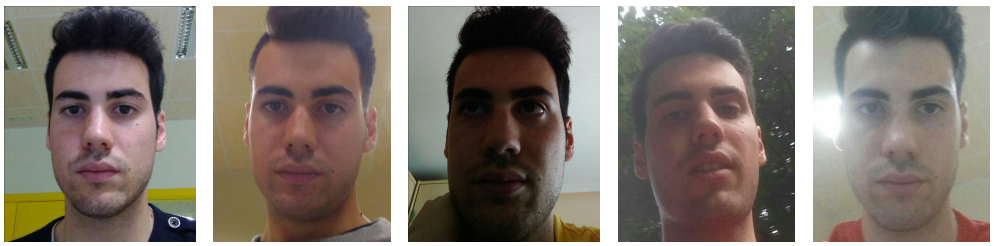


Figura 6.5: Exemplos de imaxes clasificadas como mostras.

Na Figura 6.6 vemos, pola contra, capturas que foron descartadas como mostras. Nestas imaxes non hai lugar a dúbida sobre a razón do seu rexeitamento. Existe unha minoría delas onde nin tan sequera aparece o usuario. En moitas outras si que sae, pero cortado e desenfocado debido normalmente a un movemento brusco efectuado xusto no momento en que o sistema toma a decisión de realizar

a captura. Tamén existen outras nas que o rostro non se distingue ben por saír a contraluz ou nun ambiente moi pouco iluminado.

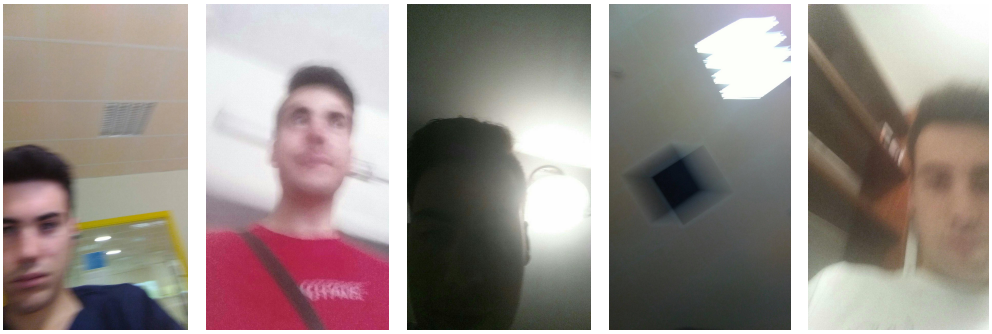


Figura 6.6: Exemplos de imaxes descartadas con mala calidade.

Finalmente, a Figura 6.7 contempla outros exemplos de capturas que foron descartadas. Sen embargo, neste caso os motivos do rexeitamento son cuestionables. A partir destas imaxes pódense obter sen problema os trazos biométricos do usuario. En ningún caso a calidade da imaxe se pode comparar á de calquera outra escollida como mostra, pero de seguro moitas destas capturas poderían formar parte do banco de mostras. As razóns principais do rexeitamento soen ser o *blur* que presentan ou o pequena que sae a cara en relación ao tamaño do cadro debido á excesiva distancia entre o usuario e o dispositivo.

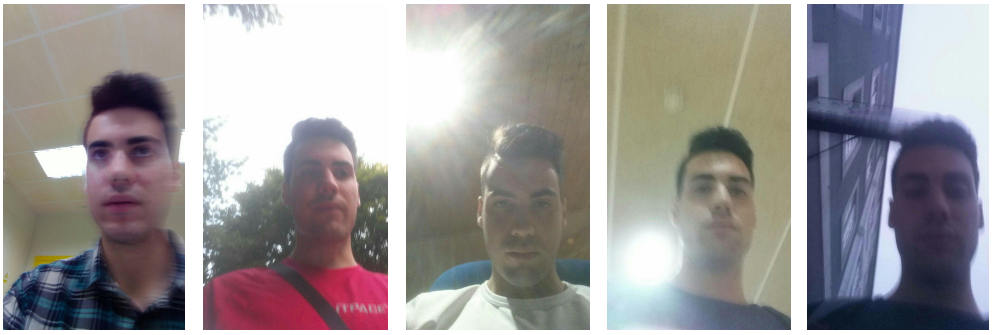


Figura 6.7: Exemplos de imaxes descartadas que poderían servir como mostras.

En definitiva, durante esta beta comprobouse que cun adestramento esixente e conservador se pode obter practicamente unha mostra do usuario por cada dúas imaxes tomadas, pero poderíase aplicar un adestramento menos restritivo para pasar a integrar no modelo imaxes como as da Figura 6.7.

Os resultados obtidos co dispositivo de desenvolvemento, o Aquaris E5s, exclúense das cifras anteriores por diversas razóns. En primeiro lugar, este móbil empregouse para probas, onde moitas veces se forzaron situacións de éxito ou fracaso que condicionarían os resultados. Por outra parte, o contexto das imaxes tomadas por

este dispositivo foi sempre o do entorno de traballo: o laboratorio *EmprendeLab* do CiTIUS. Pese a todo, podemos comentar os resultados obtidos co E5s para comparalos cos dos voluntarios da beta. En total realizáronse 548 capturas, das cales 403 pasaron a ser mostras válidas, co cal a taxa de acertos acadou un 73,54 %. Estes valores son moito mellores á media, pero hai que ter en conta a influenza dos factores anteriores. En calquera caso é de esperar que estas cifras sexan mellores, pois compre recordar que foi neste dispositivo onde se realizou o adestramento que logo se integrou na beta para ser empregado por defecto nos móbiles dos voluntarios.

Como era previsible, durante esta beta xurdiron problemas e manifestáronse fallos na aplicación. Estes aparecen documentados xunto a aqueles detectados coas probas aleatorias no informe de incidencias da seguinte sección `[INC-5, INC-6]`. Para a súa detección e seguimento empregouse a Play Console. A Figura 6.8 amosa o panel de bloqueos en tempo real producidos nos últimos 14 días para todas as versións de Android e de FaceSampler.

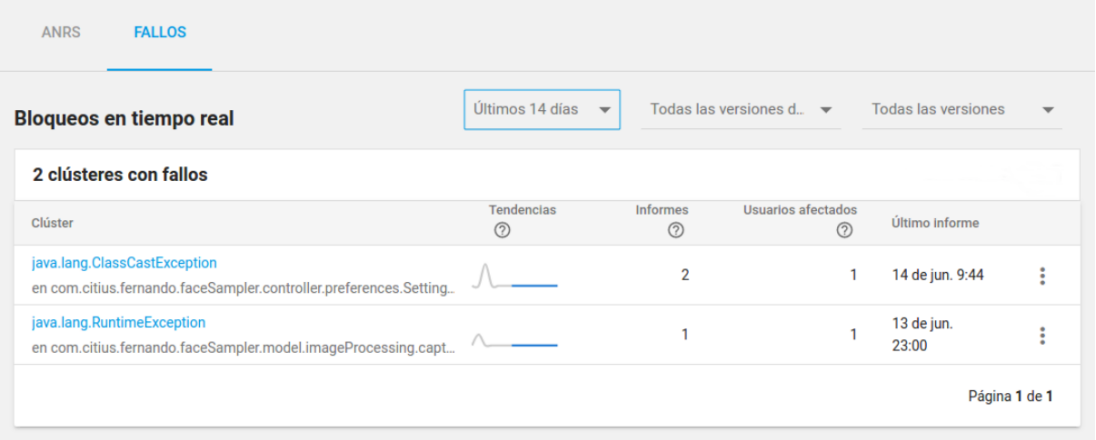


Figura 6.8: Seguemento de fallos desde a Play Console.

Para cada fallo, a consola ofrécenos cantidade de detalles, como o número de usuarios afectados, os datos dos dispositivos, súa traza de erro, etc. Na Figura 6.9 podemos ver un exemplo.

6.3. Informe de incidencias

Preséntanse a continuación as incidencias reportadas durante a execución das probas e a beta privada. A escala empregada para cuantificar a gravidade das mesmas é a seguinte:

- **Baixa**. Trátase dun fallo pouco común ou cun impacto baixo de cara á

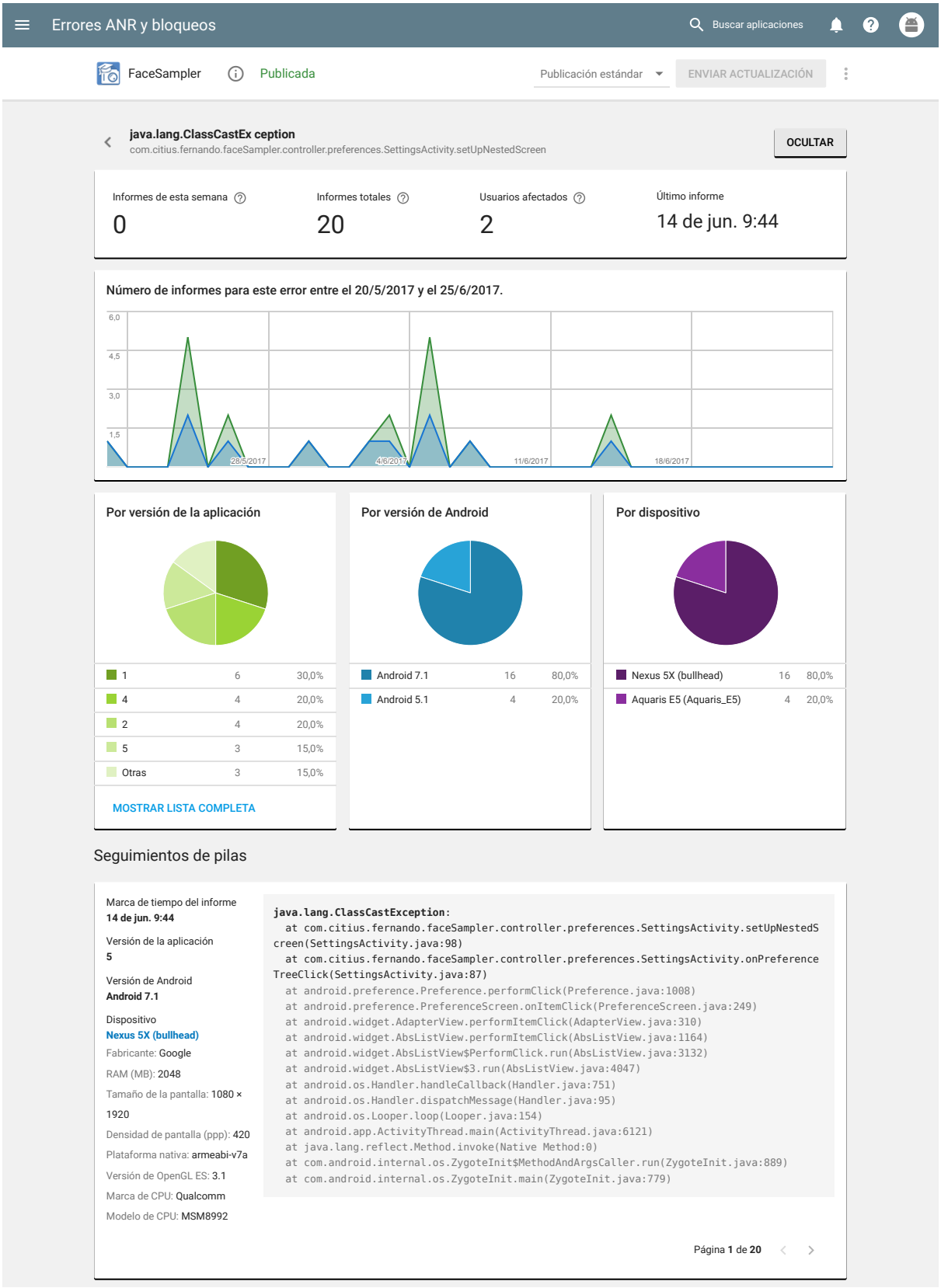


Figura 6.9: Detalles dun fallo desde a Play Console.

explotación da ferramenta, afectando a funcionalidades secundarias da aplicación, polo que a súa corrección pode esperar.

- **Media**. A súa repercusión é máis elevada que no caso anterior, podendo afectar a funcionalidades importantes da aplicación. Non se remata de comprender a orixe do fallo, polo que non hai garantías de que non se repita de novo. Compre realizar un estudo en detalle do problema.
- **Alta**. Descoñécese por completo a orixe do problema ou sábese que pode afectar a funcionalidades críticas en gran número de dispositivos. Débese abordar o problema canto antes.

Incidencias

`INC-1` **Mala predición con SVM**	
Reporta:	Fernando Estévez (desenvolvedor)
Data de detección:	10/05/2017
Gravidade:	Baixa
Descrición:	Executando as probas estruturais relativas á verificación da boa predición con SVM `CP-9` detectouse que este era máis impreciso que k-NN por algún motivo.
Accións tomadas:	En primeira instancia realizouse unha depuración da clase Java asociada a este algoritmo. Tras non detectar nada, asumiuse estar empregando mal as funcións da librería OpenCV relativas a SVM e realizouse unha proposta de cambio que será abordada nun futuro *sprint*.

`INC-2` **Fallo dos casos de proba** `CP-16` **e** `CP-17`	
Reporta:	Fernando Estévez (desenvolvedor)
Data de detección:	10/05/2017
Gravidade:	Baixa
Descrición:	Executando as probas estruturais relativas á permutación entre os modos de execución, os casos de proba `CP-16` e `CP-17` fallaron. En ambos, o motivo do fallo foi a imposibilidade de simular o evento desexado (`USER_PRESENT` no `CP-16` e `SCREEN_OFF` no `CP-17`) por tratárense de eventos do sistema e non ter permisos para tal acción.
Accións tomadas:	Asumiuse como suficiente a cobertura acadada coas probas funcionais.

`INC-3` **Bloqueo en Android 4.4**	
Reporta:	Fernando Estévez (desenvolvedor)
Data de detección:	18/05/2017

`INC-3` **Bloqueo en Android 4.4** *(cont.)*	
Gravidade:	Baixa
Descrición:	Fallou a execución da proba aleatoria `PA-1`, que debía realizarse sobre un dispositivo con versión Android 4.4. (KitKat).
Accións tomadas:	Ningunha. Previíase o fallo, posto que algúns dos compoñentes da aplicación, como o servizo de captura de imaxes ou algunhas das vistas non soportan versións da API de Android inferiores á 21 (versión 5.0 do sistema operativo). Esta proba fora planificada para ver o comportamento do sistema en versións inferiores ao mínimo recomendado (5.0).

`INC-4` **Bloqueo durante o adestramento**	
Reporta:	Fernando Estévez (desenvolvedor)
Data de detección:	18/05/2017
Gravidade:	Media
Descrición:	Executando a proba aleatoria `PA-9` recolleuse unha excepción do tipo `java.lang.NullPointerException` ao tratar de acceder á información do acelerómetro.
Accións tomadas:	Compre revisar o fallo en detalle. Realizouse unha proposta de cambio que será abordada nun futuro *sprint*.

`INC-5` **Bloqueo no panel de preferencias, categoría *Timings***	
Reporta:	Carlos Vázquez Regueiro (cliente)
Data de detección:	22/05/2017
Gravidade:	Alta
Descrición:	Durante a beta privada, o dispositivo do usuario (Nexus 5X e versión de Android 7.1) bloqueouse ao tratar de acceder á categoría *Timings* do panel de preferencias. Observando a traza do fallo desde a Play Console descubriuse que se trataba dun problema de *casteo* de clases (`java.lang.ClassCastException`).
Accións tomadas:	Abordouse o problema de inmediato por tratarse dun cambio menor. Resolveuse sen maior dilación.

`INC-6` **Inversión vertical da cámara frontal**	
Reporta:	Marco A. Araújo Matarazzo (*beta tester*)
Data de detección:	09/06/2017
Gravidade:	Baixa

INC-6 **Inversión vertical da cámara frontal** *(cont.)*	
Descrición:	Durante a beta privada, o usuario, con dispositivo Bq Aquaris X5 Plus e versión de Android 7.1, non logrou obter mostras do seu rostro, aínda que a aplicación tomou 40 capturas. Citouse ao individuo para *debuggear* o proceso de mostraxe, comprobando que o sensor da cámara frontal do seu dispositivo inverte verticalmente as imaxes, co cal o detector de rostros non é capaz de atopar a súa cara.
Accións tomadas:	Ningunha: asumiuse como un caso illado e pouco probable. A solución proposta foi incluír unha comprobación da orientación, de xeito que se busquen caras rotando a imaxe. Descartouse por ser pouco eficiente.

Capítulo 7

Conclusións e traballo futuro

Neste traballo presentouse unha proposta que pretende contribuír ao recoñecemento facial no contexto dos dispositivos móbiles, sen requirir da participación activa do usuario. O software implementado para elo, FaceSampler, consiste nunha aplicación para o ecosistema Android que logra realizar unha selección automática de mostras do usuario para poder empregalas nun sistema de verificación de rostros.

Durante este proxecto lográronse acadar de maneira exitosa os tres obxectivos propostos nun inicio. A ferramenta traballa de forma autónoma, sen desperdiciar os recursos do dispositivo. É capaz de tomar imaxes do usuario mentres este realiza un uso habitual do móbil e, a partir destas imaxes, discriminar entre aquelas que teñen un rostro humano e aquelas nas que non. A mostraxe é levada a cabo de maneira intelixente: mediante unha aprendizaxe supervisada, a aplicación é dotada coa capacidade de tomar decisións informadas sobre cando capturar unha imaxe. Ademais, coñece unha serie de regras que lle permiten permutar entre dous modos de traballo distintos: modo normal e aforro de enerxía.

Ata a data, obtivéronse bos resultados. Logrouse crear unha base de datos a pequena escala coas mostras das caras dos diversos voluntarios que empregaron a aplicación durante a fase beta. De media, podemos afirmar que a ferramenta obtén unha mostra por cada dúas capturas realizadas. Sen embargo, debido á configuración utilizada, a miúdo actúa de maneira demasiado conservadora, descartando moitas outras que poderían valer a pena.

A ferramenta xa pode ser integrada nun sistema maior de recoñecemento facial, de xeito que sirva como fonte de mostras dos rostros dos individuos. Agora, o obxectivo a longo prazo é facer xusto o contrario: implementar un sistema de identificación dentro da propia aplicación. Deste xeito, FaceSampler sería autosuficiente na labor de identificar ao propietario do dispositivo sen que este tivese que colaborar de ningunha maneira. Primeiro, comportaríase como ata agora,

realizando unha mostraxe do rostro do usuario ata obter un modelo robusto. Logo, funcionaría como un mecanismo de selección de mostras para a verificación da cara: eventualmente tomaríase unha nova mostra que sería contrastada co modelo previamente creado co fin de identificar ao suxeito.

Outra liña de avance é substituír a aprendizaxe supervisada por unha aprendizaxe continua por reforzo. Deste xeito, a aplicación sería capaz de aprender de forma completamente autónoma. Ademais, as regras de permutación entre modos de execución que a día de hoxe están definidas manualmente tamén poden ser aprendidas polo dispositivo.

Por suposto, non deben esquecerse aquelas solicitudes de cambio xurdidas da execución de probas, reporte de incidencias, avaliación heurística e peticións do cliente. Entre os cambios máis significativos están: corrixir o funcionamento de SVM, facer que o filtro de *blur* se comporte de xeito menos restritivo e incluír un arquivo de *log* onde se indique o motivo polo que se descartan as capturas rexeitadas. Todas estas tarefas xa están planificadas para ser levadas a cabo nos próximos *sprints*.

Apéndice A

Glosario

A.1. Siglas e acrónimos

ANR Tipo de erro "a aplicación non responde", etiquetado deste xeito en Android para abreviar.

APK Do inglés, *Android Application Package* (aplicación empaquetada de Android). É unha variante do formato JAR de Java é úsase para distribuír e instalar compoñentes e aplicacións para a plataforma Android.

CiTIUS Centro Singular de Investigación en Tecnoloxías da Información da Universidade de Santiago de Compostela.

ECS Elemento de Configuración do Software.

IDE Do inglés, *Integrated Development Environment* (entorno de desenvolvemento integrado).

POJO Acrónimo de *Plain Old Java Object*, empregado para referirse a clases Java simples que non dependen de ningún *framework en especial*.

SDK Do inglés, *Software Development Kit* (kit de desenvolvemento de software).

UML Do inglés, *Unified Modeling Language* (linguaxe unificada de modelado).

USC Universidade de Santiago de Compostela.

A.2. Definicións

Beta privada Denomínase deste xeito á fase previa á explotación normal do noso produto (xeralmente software) na que un público limitado e próximo

a nós proba o produto para dicirnos os fallos que atopa nel, así coma que quitaría, modificaría ou engadiría ao mesmo para melloralo.

Dataset Anglicismo empregado en informática para referirse a un conxunto de datos, habitualmente estruturado.

Mockear En programación orientada obxectos, denomínase así ao emprego de obxectos simulados para realizar probas unitarias de outros obxectos que requiren dos primeiros.

Product Owner Na metodoloxía Scrum, trátase da única persoa autorizada para decidir sobre as funcionalidades e características funcionais que terá o produto. É quen representa ao cliente, usuarios do software e todas aquelas partes interesadas no produto.

Scrum Master Na metodoloxía Scrum, a persoa que desempeña este rol é o líder do equipo Scrum. Trátase dun auténtico servidor neutral, que será o encargado de fomentar e instruír sobre os principios áxiles de Scrum.

Scrum Team En Scrum, o Scrum Team (ou simplemente "equipo"), é o equipo de desenvolvedores multidisciplinario, integrado por programadores, deseñadores, arquitectos, *testers* e demais, que en forma auto-organizada, serán os encargados de desenvolver el produto.

Sprint En Scrum, iteración de traballo de entre dúas semanas e un mes de duración, cuxa saída é un incremento de produto potencialmente entregable.

Sprint Retrospective En Scrum, reunión efectuada polo equipo de desenvolvemento ao remate de cada *sprint* co obxectivo de mellorar de maneira continua a súa produtividade e a calidade do produto que está desenvolvendo, analizando como foi a súa maneira de traballar durante a iteración, por que está conseguindo ou non os obxectivos e por que o incremento de produto que acaba de demostrar ao cliente era o que el esperaba ou non.

Sprint Review En Scrum, reunión informal onde o equipo presenta ao cliente os requisitos completados na iteración, en forma de incremento de produto preparado para ser entregado co mínimo esforzo, facendo un percorrido por eles o máis real e achegado posible ao obxectivo que se pretende cubrir.

Apéndice B

Documentos anexos

A Táboa B.1 recolle todos aqueles documentos que son achegados no CD baixo o directorio `/material_adicional`.

Arquivo	Descrición	Abrir con
`desenhoFaceSampler.mdj`	Arquivo con tódolos diagramas UML do deseño da aplicación.	StarUML 2.8.0
`FaceSampler_v4_6.apk`	APK (instalador) da última versión estable da aplicación.	LibreOffice Calc 5.1.6.2
`identificacionRiscos.ods`	Documento de xestión de riscos.	LibreOffice Calc 5.1.6.2
`mapaActividade1.png`	Exemplo de mapa de actividade xerado con Firebase para a proba aleatoria `PA-6`.	Calquera visor de imaxes
`mapaActividade2.png`	Exemplo de mapa de actividade xerado con Firebase para a proba aleatoria `PA-10`.	Calquera visor de imaxes
`planificacionTemporal.ods`	Documento resumo da planificación temporal.	LibreOffice Calc 5.1.6.2

Táboa B.1: Listado dos documentos anexos a esta memoria.

Apéndice C

Manual de usuario

O obxectivo deste manual é recoller as instrucións de descarga, instalación e uso da aplicación FaceSampler. Tamén se proporcionan os requisitos que se deben cumprir para poder facer un correcto emprego da mesma.

C.1. Descrición xeral

FaceSampler é unha aplicación capaz de tomar mostras de calidade da túa cara sen necesitar da túa colaboración. Para elo, a ferramenta actúa de forma intelixente, sabendo en que momento debe tomar unha foto. Ti mesmo podes adestrar á aplicación nesta toma de decisións, aínda que non é preciso, posto que xa ven adestrada por defecto. Non te preocupes pola consumo, FaceSampler está optimizada para ser eficiente e facer un bo uso dos recursos do teu dispositivo: cámara, batería, procesador, etc. Poderás consultar as mostras da túa cara desde a galería de imaxes do móbil.

C.2. Requisitos de instalación

A Táboa C.1 recolle os requisitos de instalación. Preséntanse tanto os requirimentos mínimos indispensables para que a aplicación funcione, como aqueles recomendados para garantir a mellor experiencia de usuario.

FaceSampler atópase actualmente en fase beta, restrinxindo o seu acceso a un grupo reducido de usuarios. Para poder instalar a aplicación no teu dispositivo terás que ser *beta tester*. Se xa o es podes continuar coa descarga e instalación. Se aínda non, envíanos un correo electrónico á dirección `fernando.estevez@rai.usc.es` co asunto "Quero probar FaceSampler". Nese correo deberás indicarnos

	Mínimo	Recomendado
Android	Versión 5.0 ou superior	Versión 5.0 ou superior
Almacenamento	43,62 MB libres	150 MB libres
Memoria RAM	500 MB	1 GB
Sensores	Cámara frontal	Cámara frontal, acelerómetro, xiroscopio e sensor de luz

Táboa C.1: Requisitos de instalación.

por que desexas probar a ferramenta, así como a túa conta habitual de Google para que te poidamos integrar na lista de *testers*.

C.3. Descarga e instalación

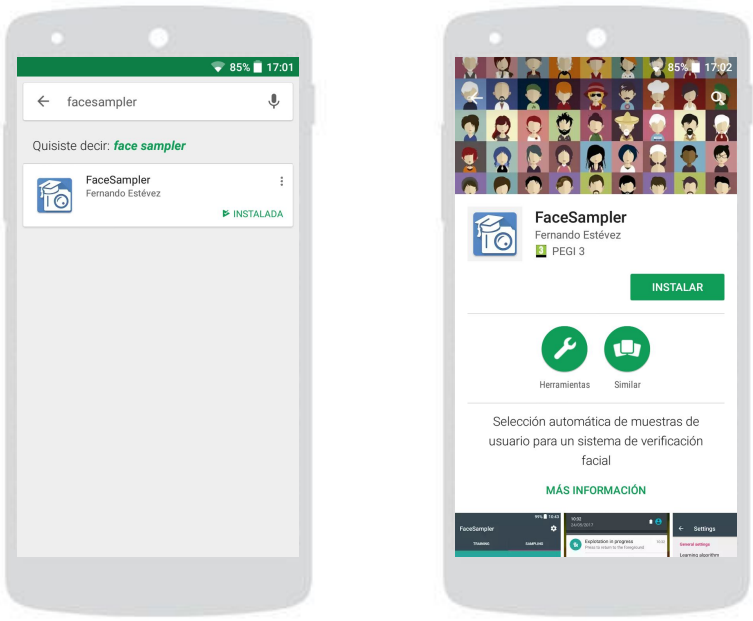
Esta aplicación é gratuíta e atópase dispoñible na Google Play Store. O primeiro que debes facer é descargala no teu dispositivo móbil. Para elo, entra na aplicación Play Store e inserta "facesampler" no cadro de busca, tal e como se amosa na Figura C.1a. O primeiro dos resultados será a nosa aplicación, preme nel para acceder á súa ficha (Figura C.1b). Tamén podes acceder directamente á ficha a través do seguinte enlace: `https://play.google.com/store/apps/details?id=com.citius.fernando.facedetector`. Lembra que se non estás acreditado como *beta tester* esta información estará oculta para ti (ler requisitos de instalación, Sección C.2).

Desde a ficha de FaceSampler terás a oportunidade de descargar e instalar a aplicación. Para elo, basta con que premas no botón *Instalar*/*Install*. De inmediato comezará a descarga e, posteriormente, instalarase a ferramenta no dispositivo de xeito automático. Nesta ficha tamén podes consultar información actualizada, como capturas de pantalla, últimas actualizacións e comentarios e valoracións dos usuarios.

C.4. Primeiros pasos

Unha vez descargada e instalada a aplicación, poderás acceder a ela pulsando na súa icona desde o menú de aplicacións do teléfono. Por defecto, amosaráseche a pantalla principal ofrecéndoche iniciar a mostraxe (Figura C.2b). Se deslizas co dedo cara a esquerda atoparás a vista de adestramento (Figura C.2a).

Se o que desexas é comezar axiña coa obtención de mostras do teu rostro, preme no botón *Comezar!*/*Start!* da vista correspondente (Figura C.2b). A partir dese momento poderás pechar ou minimizar a aplicación e realizar un uso normal



(a) Resultados da busca. **(b)** Ficha da aplicación.

Figura C.1: Descarga e instalación desde a Play Store.

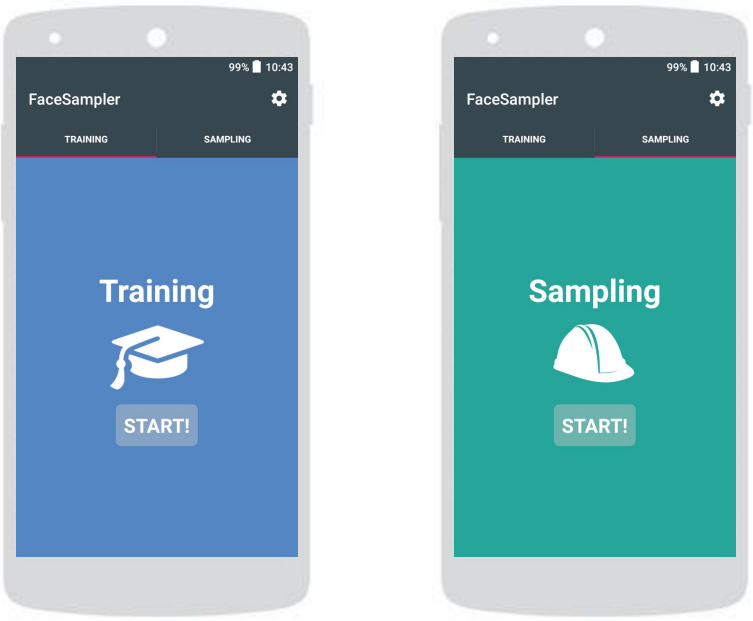
do dispositivo ata que decidas finalizar a mostraxe. Saberás que esta se está executando porque se fixará un recordatorio á barra de notificacións. Para deter o proceso, bastará con que premas *Parar!*/*Stop!*, ben desde a mesma vista ou desde o cadro de notificacións. Ten en conta que estas son as únicas dúas maneiras de deter o proceso, non servirá que peches a aplicación desde o xestor de tarefas nin que reinicies o dispositivo.

Se decides adestrar a aplicación, entón preme no botón *Comezar!*/*Start!* da vista de adestramento (Figura C.2a). FaceSampler tomará imaxes de seguido e analizará a información do contexto. Ti poderás indicarlle que situacións son as máis axeitadas para obter unha mostra: pose, movemento, iluminación, interacción, etc.

C.5. Preferencias

Desde a vista principal, se premes na icona situada na esquina superior dereita, con forma de engrenaxe, accederás ao panel de preferencias. Desde este poderás configurar a aplicación ao teu antollo. Estas son as opcións dispoñibles:

- **Preferencias da aprendizaxe/Leaning preferences**. Categoría que inclúe varias opcións relativas á aprendizaxe do dispositivo:



(a) Modo adestramento. **(b)** Modo mostraxe.

Figura C.2: Vista principal de FaceSampler.

Algoritmo de aprendizaxe/Learning algorithm. Permite elixir a estratexia de aprendizaxe a empregar. Recoméndase a opción por defecto: k-NN.

Adestramento por defecto/Default training. Permite escoller entre empregar o adestramento por defecto ou o teu propio.

Temporización/Timings. Conxunto de opcións relativas á velocidade dos procesos de adestramento e mostraxe: períodos de execución, tempo de espera tras obter unha mostra, etc.

Factores da aprendizaxe/Learning factors. Conxunto de variables do contexto a ter en conta na aprendizaxe. Poden activarse e desactivarse ao gusto do usuario, aínda que aquelas que sexan vitais estarán bloqueadas.

- **Notificacións/Notifications**. Categoría que permite controlar as distintas notificacións da ferramenta. Por agora déixase activar e desactivar as notificacións sonoras.
- **Axuda/Help**. Categoría na que se indica a versión actual da aplicación e se proporciona un enlace á súa ficha na Play Store para consultar as últimas novidades.

Apéndice D

Prototipo de detector de rostros en Java

`OpenCV` ten interfaces para as linguaxes `C`, `C++`, `Python`, `Java` (incluíndo `Android`) e `MATLAB`. Neste caso, escolleuse realizar a implementación en `Java` con vistas a que no futuro se traballaría sobre `Android`. Empregouse a versión 3.1 de `OpenCV` polo feito de ser a máis actual a día de hoxe e traballouse sobre o IDE `IntelliJ IDEA 2016.1.3` sobre o sistema operativo `Ubuntu 16.04`.

A aplicación emprega `JavaFX` para a súa interface gráfica. A Figura D.1 amosa unha captura da mesma onde se está a detectar o rostro do usuario. Para conectarse á *web cam* emprégase unha variable de tipo `VideoCapture` da biblioteca de `OpenCV`. Os *frames* obtéñense e manipúlanse como variables de tipo `Mat`.

Para a detección de caras probáronse as distintas cascadas de Haar que proporciona `OpenCV`. Optouse por importar o arquivo `haarcascade_frontalface_alt2.xml` por medio do obxecto de detección `CascadeClassifier`, posto que, tras probar as distintas posibilidades, semellou ser a máis precisa, obtendo a taxa de acertos máis alta. Para mellorar o resultado, a imaxe convértese a escala de grises e ecualízase o seu histograma. Tamén se proporciona un tamaño mínimo para a área que debe ocupar o rostro, considerándose o 15 % do tamaño total da imaxe.

A taxa de acertos media foi do 80.31 %. Este non é un dato preciso, pero permítenos ter un punto de partida. Para obter esta cifra fíxose o reconto das imaxes totais e de aquelas nas que se detectara o rostro do usuario en sucesivas probas. O acerto é, pois, detectar para cada imaxe un único rostro. Das probas inicialmente realizadas sobre a aplicación puidéronse extraer as seguintes conclusións:

- Cunha iluminación media, pódese facer un perfecto seguimento do rostro, alcanzando facilmente unha precisión do 100 % mantendo unha posición frontal do rostro e realizando movementos pouco bruscos.

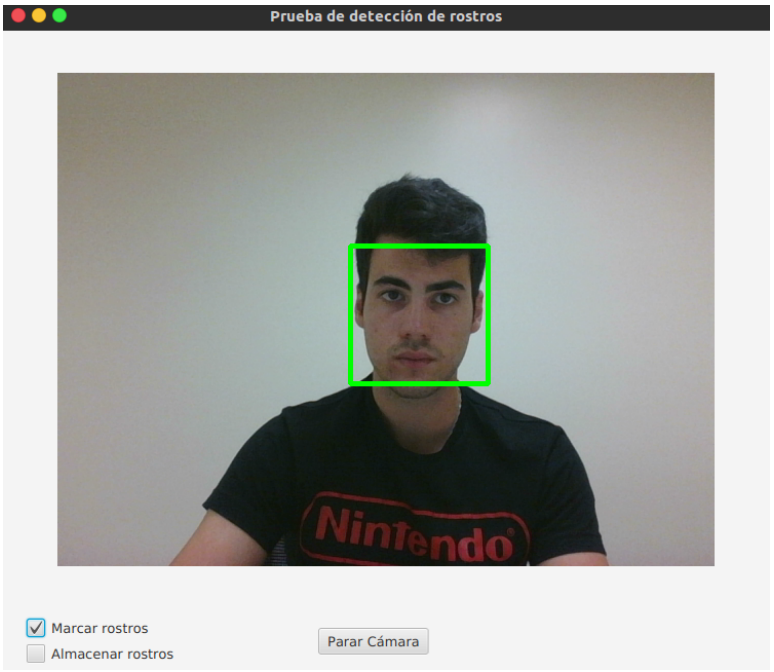


Figura D.1: Captura da interface gráfica do prototipo de detector en Java.

- Os movementos bruscos cara os laterais do cadro (esquerda e dereita) e en profundidade (cara a fronte e cara atrás) son os que menos prexudican a precisión medida.
- Cando se rota a cabeza no eixe horizontal do cadro (mirar cara arriba ou cara abaixo) obtense unha caída considerable na frecuencia coa que se detecta o rostro, pero acepta ángulos de inclinación bastante elevados.
- Ao rotar a cabeza sobre o eixe vertical (mirar á esquerda ou á dereita), os resultados son semellantes aos do punto anterior. O grao de rotación alcanzado é aceptable. Poderíamos dicir, en base ao observado, que a aplicación comeza a fallar cando non é capaz de recoñecer os dous ollos no rostro, o cal sucede ao xirar demasiado a cabeza, estando practicamente de perfil.
- O peor dos resultados prodúcese, sen lugar a dúbidas, cando se rota a cabeza sobre o eixe perpendicular ao cadro (inclinala cara un lado ou cara o outro). O limiar de inclinación permitido para que se logre identificar o rostro é mínimo.
- Compre mencionar a maiores os falsos positivos que se producen de xeito moi pouco frecuente que identifican un rostro nalgunha parte aleatoria do corpo do individuo, xeralmente no pescozo ou no peito.

Apéndice E

Estudo da precisión dos métodos usados

O presente estudo recolle unha avaliación dos factores e medidores de calidade da imaxe para unha boa detección facial. Sabemos que existen moitos medidores de calidade da imaxe orientados ao recoñecemento facial [35], que se poden clasificar en dúas grandes categorías, xenéricos e específicos:

- Os **medidores xenéricos** son aqueles que se poden aplicar en calquera modalidade biométrica. Destacamos o contraste, o brillo, o enfoque, a nitidez e a iluminación.
- A maiores existen **medidores específicos** deseñados para abordar cuestións relacionadas cunha modalidade biométrica concreta, como poderían ser os rostros, as pegadas dactilares, o iris...

E.1. Medidores xenéricos

Os medidores xenéricos semellan os principais índices a ter en conta con vistas á identificación de rostros nun futuro. Comentemos un a un:

- **Contraste**. Existen dúas solucións bastante convincentes. A primeira consiste en aplicar o segundo método contemplado en [35]. A segunda é considerar a entropía da imaxe como un medidor do contraste da mesma.
- **Brillo**. A mellor solución parece ser traballar coa imaxe en formato HSV en lugar de RGB e calcular o histograma da canle V.
- **Enfoque e nitidez**. Unha boa idea sería, no seu lugar, identificar o contrario, é dicir, o *blur* ou grado de desenfoque da imaxe, empregando para

iso a variación de Laplacian [36].

- **Iluminación**. Cando traballemos sobre un dispositivo móbil, poderemos obter unha cantidade de iluminación medida en *lux* grazas ao sensor de luz que xa tódolos *smartphones* integran, sen ter que realizar cálculos sobre a imaxe.

E.2. Medidores específicos

Os medidores específicos máis importantes no recoñecemento facial pódense dividir en tres subgrupos:

- A calidade e formato da imaxe dixital (resolución e contraste).
- Os factores dependentes da escena, entre os que destacan a rotación da cabeza, a iluminación, a apertura dos ollos, o uso de lentes e a apertura da boca.
- A exposición, o brillo o enfoque e a nitidez.

A calidade da imaxe

Este factor non semella ser algo limitante. Debido á tendencia actual dos denominados *selfies* e da mellora constante das especificacións dos *smartphone*, a maioría dos dispositivos inclúen unha cámara frontal de 1 ou 2 MP como mínimo, sendo cada vez máis habituais os 5 ou 8 MP en gamas medias.

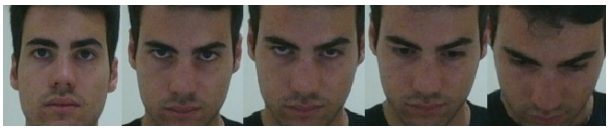
Seguemento do rostro (*tracking*) con rotación da cabeza

Realizáronse probas co prototipo Java do Apéndice D almacenando rostros en batería co obxectivo de proporcionar uns rangos acoutados de *tracking*. A Figura E.1 amosa o movemento de rotación no eixe horizontal con respecto da imaxe. Obtéñense bos resultados tanto ao mirar cara arriba (E.1a) como cara abaixo (E.1b), sendo o "fotograma limiar" o situado máis a dereita en ambos casos.

Na Figura E.2, pódese observar o rango da rotación sobre o eixe vertical. Neste caso o grao de rotación é algo máis limitado. É lóxico que sexa difícil facer un seguimento posto que nos limiares o rostro está practicamente de canto.

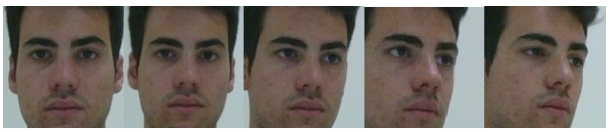
Por último, a Figura E.3 amosa o xiro sobre o eixe perpendicular. O fotograma central é o punto de partida e os extremos son os límites do rango. Este caso é o máis problemático. Unha solución rápida pero ineficiente sería procesar a imaxe tornándoa en diferentes ángulos progresivamente ata identificar o rostro.

(a) Rotación cara arriba

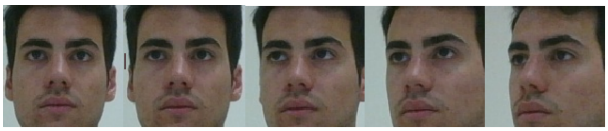


(b) Rotación cara abaixo

Figura E.1: Rotación sobre o eixe horizontal do cadro



(a) Rotación cara a esquerda



(b) Rotación cara a dereita

Figura E.2: Rotación sobre o eixe vertical do cadro

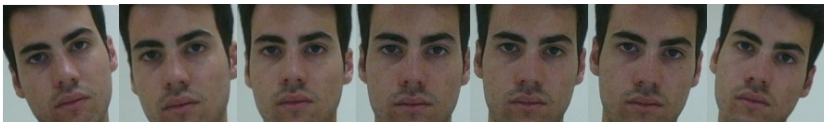


Figura E.3: Rotación sobre o eixe perpendicular ao cadro

Bibliografía

[1] B. F. Klare *et al.*, "Pushing the frontiers of unconstrained face detection and recognition: IARPA Janus Benchmark A," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1931–1939, 2015.

[2] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Tech. Rep. UM-CS-2007-049, University of Massachusetts, Amherst, 2007.

[3] V. Jain and E. Learned-Miller, "FDDB: A Benchmark for Face Detection in Unconstrained Settings," Tech. Rep. UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[4] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 529–534, 2011.

[5] C. McCool *et al.*, "Bi-modal person recognition on a mobile phone: using mobile phone data," in *Proceedings of the 2012 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pp. 635–640, 2012.

[6] J. R. Beveridge *et al.*, "The challenge of face recognition from digital point-and-shoot cameras," in *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on Biometrics*, pp. 1–8, IEEE, 2013.

[7] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[8] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[9] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning: with applications in R*, vol. 103. Springer Science & Business Media, 2013.

[10] H. Kniberg, *Scrum and XP from the Trenches: How we do Scrum*. C4Media, 2015.

[11] H. Cunningham *et al.*, "Manifesto for Agile Software Development." `http://agilemanifesto.org/`, Maio 2017.

[12] C. S. Snyder, "A Guide to the Project Management Body of Knowledge: PMBOK Guide," Project Management Institute, 2014.

[13] R. Somasundaram, *Git: Version control for everyone*. Packt Publishing Ltd, 2013.

[14] Vitae Consultores, "Guía salarial do sector TIC en Galicia, 2015-2016." `http://documents.tips/download/link/guia-salarial-sector-ti-galicia-2015-2016`, Xuño 2017.

[15] Universidade de Santiago de Compostela, "Acta do Consello de Goberno da USC, sesión do 11/11/2008." `http://imaisd.usc.es/ftp/oit/documentos/591_gl.pdf`, Xuño 2017.

[16] Agencia Estatal, "Resolución do 13 de febreiro de 2017 da Dirección General de Empleo (BOE núm. 46, de 23 de febreiro de 2017, páxs. 12769 a 12793)." `https://www.boe.es/diario_boe/txt.php?id=BOE-A-2017-1868`, Xuño 2017.

[17] Google Support, "Run tests with Firebase Test Lab for Android." `https://support.google.com/firebase/answer/6386654?hl=en`, Xuño 2017.

[18] Android Developer Documentation, "Google Play Console." `https://developer.android.com/distribute/console/index.html`, Xuño 2017.

[19] Android Developer Documentation, "Building Instrumented Unit Tests." `https://developer.android.com/training/testing/unit-testing/instrumented-unit-tests.html`, Xuño 2017.

[20] S. O. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," *Technical Report, University of Bristol (UK)*, vol. 25, 2010.

[21] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *Proceedings of the IEEE International Conference on Rehabilitation Robotics (ICORR 2011)*, pp. 1–7, IEEE, 2011.

[22] Android Developer Documentation, "Processes and Threads." `https://developer.android.com/guide/components/processes-and-threads.html`, Xuño 2017.

[23] Android Developer Documentation, "AsyncTask." `https://developer.android.com/reference/android/os/AsyncTask.html`, Xuño 2017.

[24] Android Developer Documentation, "App Manifest." `https://developer.android.com/guide/topics/manifest/manifest-intro.html`, Maio 2017.

[25] Z. Li, "Application of MVC pattern in data middleware," *Computer Engineering*, vol. 36, no. 9, pp. 70–72, 2010.

[26] Android Developer Documentation, "Activity." `https://developer.android.com/reference/android/app/Activity.html`, Maio 2017.

[27] Android Developer Documentation, "LayoutInflater." `https://developer.android.com/reference/android/view/LayoutInflater.html`, Maio 2017.

[28] Android Developer Documentation, "Material Design for Android." `https://developer.android.com/design/material/index.html?hl=es`, Maio 2017.

[29] Google, "Material Design Guidelines." `https://material.io/guidelines/material-design/introduction.html`, Maio 2017.

[30] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 249–256, ACM, 1990.

[31] Android Developer Documentation, "Fragment." `https://developer.android.com/guide/components/fragments.html`, Xuño 2017.

[32] Android Developer Documentation, "Service." `https://developer.android.com/reference/android/app/Service.html`, Xuño 2017.

[33] Microsoft Developer Network, "Publish/Subscribe." `https://msdn.microsoft.com/en-us/library/ff649664.aspx`, Xuño 2017.

[34] Android Developer Documentation, "Intent." `https://developer.android.com/reference/android/content/Intent.html`, Xuño 2017.

[35] A. Abaza, M. A. Harrison, and T. Bourlai, "Quality metrics for practical face recognition," in *Proceedings of the 21st International Conference od Pattern Recognition (ICPR, 2012)*, pp. 3103–3107, IEEE, 2012.

[36] A. Rosebrock, "Blur detection with OpenCV." http://www.pyimagesearch.com/2015/09/07/blur-detection-with-opencv/, Xullo 2016.