



UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA

Departamento de Enxeñería de Computadores

Proxecto de Fin de Carreira de Enxeñería Informática

**Caronte: plataforma para o guiado no interior
de museos**

Autor: Daniel Pan Farto

Titor: Carlos Vázquez Regueiro

Culleredo, 18 de xuño do 2018

Especificación

Título: Caronte: plataforma para o guiado no interior de museos

Clase: Proxecto clásico de enxeñería

Autor: Daniel Pan Farto

Director: Carlos Vázquez Regueiro

Tribunal:

Data de lectura:

Calificación:

À miña familia.

Agradecementos

Aos meus pais e á miña irmá porque son o que son grazas a eles. A Adri, non faría o proxecto se non fose por el. Ao meu titor, Carlos, por axudarme durante todo este tempo. Aos compañeiros de facultade polos bos momentos vividos. A Alberto, Minia, Édgar e aos meus amigos de toda a vida que me apoiaron á súa maneira.

Resumo

Poucos elementos tecnolóxicos entraron máis rápido na vida cotiá da xente que os teléfonos móbiles intelixentes, conseguindo facerse indispensábeis nun curto espazo de tempo. Son moitas as aplicacións que teñen no día a día das persoas, chegando a substituír por completo os elementos máis tradicionais cos que se levaban a cabo anteriormente. Entre estas aplicacións encóntrase a xeolocalización. Grazas a tecnoloxías coma o GPS (Global Positioning System) é posíbel situar un dispositivo en calquera punto do planeta cunha pequena marxe de erro, permitindo non só a localización dun elemento senón a indicación de rutas entre dous puntos.

Un dos principais problemas dos sistemas coma o GPS é a imposibilidade de seren utilizados en interiores. A localización en interiores presenta outro problema e é a precisión que se require para un correcto funcionamento, pois o erro asumíbel do GPS en exterior non sería viábel para os espazos interiores.

Nos últimos anos están aparecendo diversos sistemas de localización en interiores mediante a utilización de teléfonos móbiles intelixentes, algúns con gran precisión. Poderíanse describir como “sistemas GPS de interiores”. Non usan os sinais fornecidos polos satélites GPS, senón que utilizan outro tipo de sinais accesíbeis dende o teléfono móvil. Grazas a esta capacidade de localización en interiores é posíbel tamén a navegación.

A cantidade de información que pode haber ao redor dunha obra de arte pode ser abrumadora: dende as técnicas utilizadas para a creación dunha escultura, como o contexto histórico do artista ou o movemento artístico ao que pertence unha pintura. Cos métodos tradicionais, non sería posíbel mostrar toda a información dispoñíbel sobre unha obra nos museos.

O tempo dispoñíbel á hora de percorrer un museo de gran tamaño adoita ser un problema para os visitantes. Hai veces nas que o visitante desexa é admirar as obras de certos artistas ou movementos artísticos, o cal pode ser complicado se non posúe moita información.

Palabras clave

Android, Localización en interiores, Situm Technologies, Guiado en museos.

Índice xeral

Índice de figuras	xv
Índice de táboas	xix
1. Introdución	1
1.1. Contexto	1
1.2. Solucións actuais	2
1.3. Obxectivos	3
1.4. Estrutura da memoria	3
2. Conceptos teóricos	5
2.1. Android	5
2.2. Posicionamento	5
2.2.1. Posicionamento en exteriores: GPS	6
2.2.2. Posicionamento en interiores	6
2.3. Servizos web	6
2.3.1. Servizos web REST	7
3. Fundamentos tecnolóxicos	9
3.1. Recursos hardware	9
3.1.1. Dispositivos (teléfonos) móbiles intelixentes	9
3.2. Recursos software específicos do proxecto	10
3.2.1. SDK Situm	10
3.2.2. Apache Tomcat	11

3.2.3. Amazon Web Services	11
3.2.4. Eclipse	11
3.2.5. Java	11
3.2.6. Android Studio	12
3.2.7. Postman	12
3.2.8. SQuirreL	12
3.3. Recursos de xestión do proxecto	12
3.3.1. Git	12
3.3.2. LaTeX e TeXstudio	13
3.3.3. Inkscape	13
3.3.4. StarUML	13
4. Metodoloxía de desenvolvemento	15
4.1. Que é Scrum?	15
4.2. Roles	15
4.2.1. Roles centrais	15
4.2.2. Roles non centrais	16
4.3. Ciclo de proceso	17
4.3.1. Inicio de ciclo	18
4.3.2. Sprint planning meeting	18
4.3.3. Daily standup	18
4.3.4. Sprint review meeting	18
4.3.5. Sprint retrospective	19
4.4. Vantaxes e desvantaxes de Scrum	19
4.4.1. Vantaxes	19
4.4.2. Desvantaxes	19
4.5. Adaptación da metodoloxía	20
5. Análise	21
5.1. Actores	21
5.2. Casos de uso	22

5.2.1. Usuario Android	22
5.2.2. Xestor de contido	26
5.2.3. Administrador do sistema	29
6. Planificación e custos	31
6.1. Desenvolvemento	31
6.1.1. Sprint 1: Aplicación Android: localización dentro dun edificio	31
6.1.2. Sprint 2: Servidor inicial: POIs e percursos	32
6.1.3. Sprint 3: Lectura datos dende aplicación Android	33
6.1.4. Sprint 4: Autenticación de usuarios	34
6.1.5. Sprint 5: Edición de POIs e percursos dende a aplicación Android	35
6.1.6. Sprint 6: Valor engadido da localización Situm	36
6.1.7. Sprint 7: Imaxes para POIs	37
6.2. Planificación e custos iniciais	37
6.2.1. Planificación inicial	38
6.2.2. Custo inicial	38
6.2.3. Análise de riscos e plans de continxencia	40
6.3. Planificación e custos finais	42
6.3.1. Planificación final	42
6.3.2. Incidencias e resolución	43
6.3.3. Custo final	44
7. Deseño	47
7.1. Arquitectura xeral	47
7.2. Modelo de datos	49
7.2.1. EDIFICIO	49
7.2.2. USUARIO	50
7.2.3. USUARIO_EDIFICIO	51
7.2.4. PUNTO_INTERESE	51
7.2.5. PERCORRIDO	52
7.2.6. PERCORRIDO_PUNTO_INTERESE	53

7.2.7. CONTA_SITUM	54
7.2.8. CONTA_SITUM_USUARIO	54
7.2.9. IMAXE	55
7.3. Servidor	55
7.3.1. División en capas	56
7.3.2. Transaccionalidade	57
7.3.3. API REST	58
7.3.4. Intercambio de datos	60
7.3.5. Securización das comunicacións	61
7.4. Sistema de autenticación a través de Google	62
7.5. Aplicación Android	63
7.5.1. Actividades	64
7.5.2. Comunicación co servidor	66
7.5.3. Comunicación con Situm	66
7.5.4. Permisos necesarios	67
8. Implementación	69
8.1. Servidor	69
8.1.1. Acceso á base de datos	69
8.1.2. Transaccionalidade	72
8.1.3. Construcción dos servizos web	73
8.1.4. Autorización nos servizos	75
8.2. Aplicación Android	75
8.2.1. Solicitud de permisos	75
8.2.2. Acceso a Situm	77
8.2.3. Acceso ao servidor	80
8.3. Autenticación	82
9. Probas	87
9.1. Probas de validación de cada sprint	87
9.2. Consola de Google	88

9.3. Probas de rendemento	89
9.4. Enquisa usuarios	89
10. Conclusóns e traballo futuro	91
10.1. Conclusóns	91
10.2. Traballo futuro	92
A. Casos de uso	95
B. Instalación do sistema Caronte	107
B.1. Requisitos	107
B.1.1. Servidor Caronte	107
B.1.2. Aplicación Android: Caronte	107
B.2. Instalación	108
B.2.1. Servidor Caronte	108
B.2.2. Aplicación Android: Caronte	108
B.3. Manuales de usuario	108
B.3.1. Administrador do servidor Caronte	108
B.3.2. Aplicación Caronte para usuarios anónimos	108
B.3.3. Aplicación Caronte para xestores de contidos	109
C. Localización en interiores de Situm	111
C.1. Nova conta en Situm	111
C.2. Dashboard de Situm	111
C.3. Editar edificios en Situm	112
C.4. Calibración de cada planta	112
C.5. Punto de interese Situm	113
C.6. Definición de rutas	114
D. Contido do DVD adxunto	117

Índice de figuras

4.1. Proceso de desenvolvemento en Scrum.	17
5.1. Casos de uso do Usuario Android dentro da pantalla inicial.	23
5.2. Casos de uso referentes á localización do Usuario Android.	24
5.3. Casos de uso referentes á visualización de POIs e percorridos.	25
5.4. Casos de uso do Xestor de contido referentes aos POIs.	27
5.5. Casos de uso do Xestor de contido referentes aos percorridos.	29
5.6. Casos de uso do Xestor de contido referentes aos percorridos.	30
6.1. Diagrama de Gantt da planificación inicial para os sprints.	38
6.2. Diagrama de Gantt para os sprints despois de realizar o seguemento ao proxecto.	43
7.1. Arquitectura xeral da plataforma Caronte para a guía de museos.	48
7.2. Modelo de datos da plataforma Caronte.	50
7.3. Diagrama de clases coas entidades.	56
7.4. Diagrama de clases cos DAOs.	57
7.5. Exemplo dunha solicitude ao servizo web.	60
7.6. Exemplo dunha resposta do servizo web.	61
7.7. Exemplo dunha cabeceira REST securizada.	61
7.8. Diagrama de secuencia da autenticación a través de Google.	62
7.9. Chave para o acceso a Google Maps.	65
8.1. Configuración da base de datos no ficheiro XML.	70

8.2. Clase de implementación dun DAO.	70
8.3. Inxección dunha consulta nun DAO.	71
8.4. Método de consulta dun DAO.	71
8.5. Construcción dun VO dentro do DAO.	71
8.6. Exemplo dunha consulta en SQL.	72
8.7. Método de inserción dun DAO.	72
8.8. Configuración da transaccionalidade no ficheiro XML.	73
8.9. Configuración da transaccionalidade dun método onde se modifican datos.	73
8.10. Configuración da transaccionalidade dun método de só lectura.	73
8.11. Exemplo de etiquetas sobre o servizo.	74
8.12. Exemplo de método Get con parámetro do servizo.	74
8.13. Exemplo de método Post do servizo.	74
8.14. Exemplo de método Delete do servizo.	75
8.15. Configuración da autenticación básica no servidor.	76
8.16. Permisos necesarios na aplicación Android.	76
8.17. Comprobación e solicitude de permisos necesarios na aplicación Android.	77
8.18. Inicialización do SDK de Situm.	78
8.19. Creación do location manager de Situm.	78
8.20. Activación do posicionamiento de Situm.	79
8.21. Chamada para a recuperación dos niveis dun edificio en Situm.	80
8.22. Construcción dunha clase de acceso ao servidor de Caronte.	80
8.23. Método para a realización dunha chamada ao servidor de Caronte.	81
8.24. Método para o tratamiento de datos devoltos por unha chamada ao servidor de Caronte.	82
8.25. Código para a autenticación con Google na aplicación.	83
8.26. Código para a autenticación con Google na aplicación (2).	84
8.27. Código para a autenticación con Google na aplicación (e 3).	85
9.1. Pantalla coas incidencias na consola de Google Play.	88
C.1. Vista inicial do dashboard de Situm.	112

C.2. Vista de modificación dun edificio.	113
C.3. Aplicación de calibración: Situm Mapping Tool.	115
C.4. Proceso de creación de POIs en Situm.	116
C.5. Proceso de creación de rutas.	116

Índice de táboas

3.1. Características One Plus 3T	10
6.1. Horas planificadas para o traballador.	39
6.2. Custos planificados en recursos humanos.	39
6.3. Custos en recursos materiais.	40
6.4. Custos totais estimados.	40
6.5. Avaliación dos riscos do proxecto.	41
6.6. Horas reais para o traballador.	45
6.7. Custos finais en recursos humanos.	45
6.8. Custos totais unha vez finalizado o proxecto.	45
7.1. Descripcións da API Rest para o tratamiento de imaxes.	58
7.2. Descripcións da API Rest para o tratamiento da información xeral.	59
9.1. Resultados da enquisa a usuarios.	90
A.1. Caso de uso Autenticarse.	95
A.2. Caso de uso Logout.	95
A.3. Caso de uso Recuperar contas Situm.	96
A.4. Caso de uso Acceder mapa.	96
A.5. Caso de uso Amosar mapa edificio.	96
A.6. Caso de uso Cambiar piso.	97
A.7. Caso de uso Localización no interior dun edificio.	97
A.8. Caso de uso Amosar lista de POIs dun edificio.	97

A.9. Caso de uso Amosar POI no mapa.	98
A.10.Caso de uso Amosar información dun POI.	98
A.11.Caso de uso Guiar a POI.	98
A.12.Caso de uso Amosar lista de percursos dun edificio.	99
A.13.Caso de uso Amosar percurso no mapa.	99
A.14.Caso de uso Guiar a través de percurso.	99
A.15.Caso de uso Amosar información dun percurso.	100
A.16.Caso de uso Amosar lista de imágenes dun POI.	100
A.17.Caso de uso Amosar imaxe dun POI.	100
A.18.Caso de uso Crear POI.	101
A.19.Caso de uso Modificar POI.	101
A.20.Caso de uso Eliminar POI.	101
A.21.Caso de uso Estimar tempo POI.	102
A.22.Caso de uso Engadir imaxe.	102
A.23.Caso de uso Modificar datos imaxe.	102
A.24.Caso de uso Eliminar imaxe.	103
A.25.Caso de uso Crear percurso.	103
A.26.Caso de uso Modificar información percurso.	104
A.27.Caso de uso Eliminar percurso.	104
A.28.Caso de uso Estimar tempo percurso.	104
A.29.Caso de uso Engadir POI a percurso.	105
A.30.Caso de uso Eliminar POI de percurso.	105
A.31.Caso de uso Dar de alta un edificio.	105
A.32.Caso de uso Dar permiso a un usuario sobre un edificio.	106
A.33.Caso de uso Dar de alta unha conta Situm.	106
A.34.Caso de uso Dar permiso a un usuario sobre un edificio.	106

Capítulo 1

Introducción

Neste primeiro capítulo explicaremos brevemente os aspectos básicos do proxecto: contexto xeral do problema a solventar, solucións existentes na actualidade e a nosa proposta para o proxecto. No último apartado deste capítulo explicaremos a estrutura da memoria.

1.1. Contexto

Os dispositivos móbiles intelixentes (smartphones) revolucionaron a tecnoloxía nestes últimos anos. Pódense ter todo tipo de aplicacións e utilidades ao alcance da nosa man e gardalas nun peto, cun tamaño semellante ao que pode ter unha carteira. Grazas a estes dispositivos podemos realizar todo tipo de accións que antes estaban limitadas a unha soa máquina ou aparello: sustituíron calculadoras, libretas, lanternas, mapas... É precisamente na sustitución dos mapas onde se quere centrar este proxecto. De todos é coñecida a utilidade das aplicacións baseadas na xeolocalización para o guiado e posicionamento en lugares descoñecidos nos que pisamos por primeira vez, xa non é unha aventura viaxar a unha cidade sen coñecela previamente; o único que se precisa é un móvil intelixente e activar o servizo GPS do mesmo para poder percorrer a cidade sen se perder. Mais o GPS ten, entre outras, unha gran limitación: non poden ser utilizados no interior de edificios. Esta capacidade tamén sería moi útil se se puidese utilizar en certos edificios cun gran tamaño e ao que un posíbel usuario non estea acostumado.

Recentemente, producíronse grandes avances na determinación fiábel e precisa da posición dun teléfono móvil en interiores, polo que xa se pode solventar esa restrición do GPS.

Son múltiples os casos nos que pode resultar útil un sistema de guiado nun edificio, xa sexa público ou privado. Entre eles podemos destacar centros médicos, nos que poder guiar a pacientes á súa consulta sen necesidade de solicitar axuda; centros comerciais, para situar as tendas do seu interior; ou museos, nos que poder ofrecer a localización das obras da súa colección. Ésta última opción foi a seleccionada para este proxecto xa que ten engadidos máis interesantes como a creación de percorridos para que os usuarios visiten o museo sengundo os seus gustos ou o tempo dispoñíbel.

Para a realización do proxecto seleccionouse o sistema de posicionamento en interiores de Situm debido ao seu bo funcionamento e á súa accesibilidade.

1.2. Solucións actuais

A continuación enumeramos distintas solucións para o guiado e obtención de información dentro de museos. Imos dende as más estendidas e básicas ata as más elaboradas cun maior grao de semellanza coa nosa idea de proxecto.

- Guías en papel: os típicos mapas con breves explicacións existentes en todos os museos. Precisanse paneis ao longo do museo para o posicionamento. Non hai ningunha interacción.
- Audioguías: configurábeis por idioma. É necesario introducir o código asignado a un elemento para escoitar un audio sobre el.
- Comezan a estenderse as guías multimedia en dispositivos intelixentes debido ás súas grandes posibilidades:
 - Solucións propias para cada museo: non teñen tantas opcións como as xenéricas pero compénzano cunha maior especialización e adaptación ao museo en cuestión. O museo do Prado é un exemplo deste tipo de solucións.

- ATS Heritage: empresa que crea guías multimedia xenéricas para museos. As guías poden estar dispoñíbeis para iOS, Android e Windows Phone. Teñen a capacidade de localizar no exterior mais non no interior.
- OrpheoGroup: fabrican dispositivos multimedia propios. O máis configurábel consiste nun teléfono móvil intelixente para o cal se pode preparar contido mediante software propio da empresa.
- AcousticGuide: elaboran hardware propio. Permite a localización en interior.

1.3. Obxectivos

O obxectivo deste proxecto é desenvolver un sistema que permita a localización e guiado dentro dun edificio (museo), cunha inversión reducida e sen infraestrutura específica, a través dun dispositivo móvil. A aplicación permitirá a localización do usuario ou de puntos de interese, e permitirá unir esos puntos segundo percorridos semanticamente relacionados.

Os obxectivos específicos son:

- Localizar un usuario dentro do edificio.
- Localizar un punto de interese dentro do edificio.
- Permitir o guiado do usuario ata un punto de interese.
- Permitir o guiado do usuario entre distintos puntos mediante percorridos.
- Proporcionar todas as ferramentas precisas para que un usuario que teña que administrar o edificio poida realizalo comodamente de xeito sinxelo.

1.4. Estrutura da memoria

A memoria divídese en dez capítulos intentando conseguir unha división detallada de cada paso da elaboración do proxecto. Os primeiros puntos serven para dar un contexto ao proxecto, sen atender aínda á súa elaboración. No capítulo de Conceptos teóricos faise un pequeno resumo explicativo dos aspectos xerais tratados no proxecto.

Conceptos coma o posicionamento en interiores serán explicados neste punto. No terceiro capítulo, Fundamentos tecnolóxicos, revisanse os elementos utilizados na creación do proxecto, tanto dispositivos hardware coma software: smartphones, entornos de desenvolvemento e aplicacións varias. No cuarto punto, Metodoloxía de desenvolvemento, explícanse as condicións nas que se elaborou o proxecto.

É a partir do quinto punto, Análise, onde se comeza a entrar no detalle do mesmo. Describense os requisitos que debe cumplir o proxecto e os casos de uso que dan conta deles. O sexto punto correspón dese coa Planificación e custos, onde se detallan os pasos que se deron na elaboracion do proxecto xunto cos custos desglosados.

No séptimo punto faise fincapé no deseño da aplicación, tanto da arquitectura de todos os sistemas involucrados, explicando os motivos das eleccións e en como están montados coma solucións; coma dos detalles da base de datos e do servizo web. No seguinte capítulo explícase o proceso de Implementación da aplicación Android e do servizo web.

Os dous últimos capítulos son os de Probas, onde se comproban as funcionalidades do sistema, e o de Conclusións e traballo futuro, onde se reflexiona sobre os obxectivos realmente logrados unha vez rematado.

Finalmente, remátase a memoria en varios apéndices.

Capítulo 2

Conceptos teóricos

Neste capítulo farase unha relación dos conceptos teóricos nos que se basea o proxecto.

2.1. Android

É un sistema operativo principalmente dirixido a dispositivos móbiles que ten como base o núcleo Linux. Nos seus inicios foi desenvolvido pola empresa Android Inc., financiada por Google, multinacional estadounidense que acabaría por se facer co seu control. É o sistema operativo con maior cuota de mercado dentro dos dispositivos móbiles cunha ampla marxe sobre a súa competencia.

Escóllese este sistema operativo por ser de código aberto, facilitar a programación sobre el e por ter unha base de usuarios maior.

2.2. Posicionamiento

O posicionamento é a acción de situar algo ou alguém nun lugar, especialmente referíndose a mapas. Distínguese entre posicionamiento en exterior e interior.

2.2.1. Posicionamento en exteriores: GPS

O Global Positioning System (GPS) ou Sistema de Posicionamento Global en galego, é un sistema de navegación por satélite utilizado en todo o mundo. Foi creado e é mantido polo goberno dos Estados Unidos, tendo unha orixe militar. Funciona grazas a 27 satélites (24 principais e 3 de respaldo) que se atopan en órbita sobre o planeta. Permiten a localización dun dispositivo en calquera punto do globo. O posicionamento lógrase mediante triangulación cando se consigue a conexión con, como mínimo, catro satélites. Non é preciso o uso de redes telefónicas para estas conexións xa que non é preciso que o usuario envíe ningún tipo de información.

Os satélites emiten un sinal de xeito continuo, mais este sinal non é moi potente polo que pode verse afectado por obstáculos, tales como dificultades orográficas ou edificios, polo cal non é posíbel utilizalo dentro de calquera tipo de edificio.

2.2.2. Posicionamento en interiores

O impedimento de utilizar o GPS en interiores provocou que se buscasen maneiras de permitir un posicionamento para dispositivos móbiles en entornos baixo teito. Para logralo utilizase calquera tipo de información recollida polo dispositivo, tales como ondas de radio, campos magnéticos ou sinais acústicos. Para unha maior efectividade destes sistemas pódense dispoñer de diversos elementos emisores de ondas en puntos estratégicos coma poden ser os beacons: dispositivos emisores de baixo consumo que utilizan a tecnoloxía Bluetooth.

É unha porción de mercado que está en disputa entre varias empresas que están crecendo, como poden ser Situm, Pole Star, Indoor Atlas, entre outras.

2.3. Servizos web

Son un conxunto de protocolos e estándares que permiten o intercambio de datos entre distintas aplicacións. A comunicación non se ve afectada polas linguaxes nas que se escriben esas aplicacións nin polas plataformas nas que estas se executan, polo que non é preciso coñecer como están feitos.

2.3.1. Servizos web REST

REST (REpresentational State Transfer - transferencia de estado figurativo) é un estilo de arquitectura software para sistemas distribuídos que require unha comunicación cliente/servidor sen estado e cacheable a través dunha interface uniforme entre compoñentes.

Neste estilo arquitectónico, os datos e as funcionalidades utilizadas son consideradas recursos e accédese a eles mediante Identificadores de Recurso Uniformes (Uniform Resource Identifiers - URI), que tradicionalmente se identifican coas ligazóns na Web. Os recursos son utilizados mediante un conxunto de operacións simples e ben definidas. A arquitectura utilizada é de tipo cliente/servidor e está deseñada para utilizar un protocolo de comunicacións sen estado, tipicamente HTTP. En REST, os clientes e os servidores intercambian representacións de recursos utilizando unha interface e un protocolo estandarizados. Os principios que persegue as aplicacións REST son a simplicidade, a lixeireza e a rapidez.

Capítulo 3

Fundamentos tecnolóxicos

Neste capítulo farase unha relación das tecnoloxías hardware e software nas que se basea o proxecto e outras que están en clara conexión. Tamén se tratará todo aquel software utilizado para a creación deste proxecto.

3.1. Recursos hardware

Para a realización deste proxecto empregáronse certos recursos hardware que pasaremos a detallar a continuación:

3.1.1. Dispositivos (teléfonos) móbiles intelixentes

Un dispositivo (teléfono) móvil intelixente é un ordenador de tamaño reducido con capacidades de comunicación pola rede telefónica a través de voz e datos. Están deseñados para realizar múltiples tarefas a través de software instalado no mesmo, xa sexa polo usuario ou fabricante, que se apoia nun sistema operativo que permite a utilización do hardware incluído no teléfono.

Os primeiros dispositivos foron creados a finais do século pasado mais non comezaron a estenderse ata esta década. Na actualidade é tan habitual o uso destes dispositivos que xa se contan por miles de millóns en todo o planeta.

Debido ao seu uso tan estendido é a opción escollida para ser a base do proxecto.

One Plus 3T

O dispositivo utilizado para as probas da aplicación Android foi o terminal One Plus 3T, coa versión 8.0 do sistema Android, un terminal de gama media que permitirá facerse unha idea do desempeño xeral da aplicación neste tipo de dispositivos.

As súas características pódense observar na táboa 3.1

	Características One Plus 3T
Sistema Operativo	Android 8.0 Oreo
Procesador	Qualcomm Snapdragon 821 Quad Core, Kryo
Dimensíóns	152,7 x 74,7 x 7,35 mm
Peso	158 g
Pantalla	5,5 pulgadas cunha resolución 1080p Full HD (1920 x 1080 píxeles)
Memoria	6 GB
Almacenamento	64 GB

Táboa 3.1: Características One Plus 3T

3.2. Recursos software específicos do proxecto

Estes recursos foron básicos na creación do sistema, xa sexa para a elaboración do código ou como plataforma para executar o noso código. A continuación pásase a explicar brevemente a súa función:

3.2.1. SDK Situm

Situm é unha empresa galega especializada no posicionamento en interiores. O seu sistema pódese implantar en calquera edificio cunha infraestrutura mínima, xa que incluso pode funcionar sen instalación de ningún tipo de dispositivos. Grazas a iso é posíbel un despregue inmediato no que só sería preciso a calibración do edificio que se pode levar a cabo en minutos. Situm proporciona un SDK (Software Development Kit - ou en galego, Kit de desenvolvemento de software) que permite utilizar o seu sistema tanto en Android coma iOS. Dispoñen dunha API pública na que poder revisar todos os elementos incluídos no SDK facilitando así o seu uso.

3.2.2. Apache Tomcat

Apache Tomcat é un contedor de servlets Java desenvolvido pola Apache Software Foundation. Implementa varias especificacións da Java Enterprise Edition e proporciona un entorno para servidores web HTTP no cal se pode executar código Java. Foi a opción escollida para o despregue dos servizos.

3.2.3. Amazon Web Services

Amazon Web Services (AWS) é unha plataforma de servizos na nube que ofrece gran potencia de cómputo, almacenamento para bases de datos, entrega de contido, entre outras funcionalidades aportando gran flexibilidade, un entorno escalábel e moi fiábel. Esta plataforma supón un gasto que depende da capacidade de cómputo requirida así coma o uso de rede. Utilizouse esta plataforma para lanzar o noso servidor e almacenar a base de datos.

3.2.4. Eclipse

Eclipse é un conxunto de ferramentas, proxectos e grupos de traballo en código aberto. Entre as ferramentas das que se compón, unha das más utilizadas é o seu IDE (Integrated Development Environment - ou en galego, Entorno de desenvolvemento integrado), dispoñíbel para distintos sistemas operativos e amplamente utilizado para a programación en Java. Este foi o entorno que se utilizou para a elaboración dos servizos web.

3.2.5. Java

A plataforma Java é un entorno de computación creado por Sun Microsystems capaz de executar aplicacións desenvolvidas na linguaxe de programación Java, principalmente. A encargada de executar as aplicacións é a súa máquina virtual xunto cunha serie de bibliotecas estándar. Foi utilizada conxuntamente coa linguaxe de programación do mesmo nome para a creación dos servizos web.

3.2.6. Android Studio

É o IDE oficial para a programación en Android. Dende a súa saída substituíu ao Eclipse que era o anterior. Ao estar específicamente deseñado para o desenvolvemento de aplicacións en Android fai que a experiencia de programación para este sistema sexa moito más rápida e cómoda. Foi o entorno utilizado para a programación da aplicación de Android.

3.2.7. Postman

Postman é un IDE destinado ao desenvolvemento de APIs. Permite a creación de peticóns a APIs e a elaboración de tests de validación do seu comportamento entre outras características. Utilizouse para as probas da parte servidora.

3.2.8. SQuirreL

SQuirreL é un cliente de SQL que utiliza Java para se executar. A súa interface de acceso á base de datos é gráfica, o que permite ver a estrutura da mesma, así como navegar a través dos seus esquemas e táboas. A súa funcionalidade pode estenderse a través de plugins. Foi a aplicación utilizada para a xestión da base de datos.

3.3. Recursos de xestión do proxecto

A continuación faise unha relación dos recursos utilizados na xestión do proxecto ou na elaboración da memoria:

3.3.1. Git

Git é un sistema de control de versións que permite a coordinación de distintas persoas que realizan cambios sobre o mesmo traballo. Está orientado á rapidez, manter a integridade dos datos e soportar modelos de traballo distribuídos e non lineais. Como servidor para o control de versións utilizouse GitHub.

3.3.2. LaTeX e TeXstudio

LaTeX é un sistema de composición de textos orientado á creación de documentos cunha alta calidade tipográfica. Polas súas características e posibilidades, úsase especialmente na xeración de artigos e libros científicos que inclúen expresións matemáticas. Editoriais científicas de primeira liña utilizan este sistema debido á súa calidade. TeXstudio é un IDE para a creación de documentos en LaTeX, cun deseño simple e agradábel, que integra numerosas ferramentas para a creación de documentos científicos. É de código aberto. Utilizouse este entorno para a elaboración da memoria.

3.3.3. Inkscape

É un editor de gráficos vectoriais, libre e de código aberto. Permite a conversión de ficheiros de imaxes clásicos en documentos de gráficos vectoriais. Utilizouse para a conversión das imaxes utilizadas na memoria.

3.3.4. StarUML

StarUML é un proxecto de código aberto que permite a creación de calquera tipo de diagrama UML. Utilizouse para a elaboración dos diagramas incluídos na memoria.

Capítulo 4

Metodoloxía de desenvolvemento

Para a realización de este proxecto seguiuse unha versión modificada e simplificada da metodoloxía Scrum, xa que ao ser un proxecto realizado por unha única persoa non se podería seguir a metodoloxía sen modificacións. Escolleuse esta metodoloxía por favorecer o desenvolvemento rápido de aplicación e por estar en auxe actualmente.

4.1. Que é Scrum?

Scrum é unha metodoloxía áxil para o desenvolvemento de proxectos, que define un conxunto de accións e roles, executada mediante iteracións despois das cales se obtén un produto entregábel ao cliente. Isto permite que o usuario revise estes produtos e que proporcione feedback durante todo o proceso. Deste xeito pode dar a súa opinión en cada iteración que permita ao equipo axustarse aos seus desexos.

4.2. Roles

Distínguense dous tipos de roles dentro de Scrum:

4.2.1. Roles centrais

Son aqueles cuxa participación é indispensábel na realización do proxecto. A responsabilidade do éxito ou o fracaso tanto de cada sprint individual coma do proxecto

global recae sobre eles.

Product owner

É o encargado de representar os desexos das partes interesadas no proxecto, polo que ten que coñecer as necesidades do cliente e ter claras as súas prioridades para definir e priorizar tarefas, xa que debe comunicárlas ao equipo de traballo.

Scrum master

Este usuario debe asegurarse de que se cumple a metodoloxía, exercendo de guía nas xuntanzas e proporcionando axuda ao equipo en caso de que se produza algúun problema. Debe exercer de muro antes presións externas, evitando que o resto do equipo sufra as presións. Estas funcións son as que levaría a cabo un xefe de proxecto nas metodoloxías clásicas, mais non é o líder do equipo xa que os seus integrantes autoorganízanse.

Equipo Scrum

Son os desenvolvedores do proxecto, os encargados de realizar as tarefas escollidas polo Product owner e os responsábeis de entregar os produtos. O número de integrantes adoita atoparse entre 5 e 9 persoas. En cada sprint realizan as tarefas propias do desenvolvemento: análise, deseño, implementación e probas; polo que deben contar con habilidades transversais que lles permitan realizar o traballo.

4.2.2. Roles non centrais

Son aqueles nos que a súa participación non é indispensábel na realización do proxecto, xa que non recae sobre eles a elaboración do proxecto.

Usuario

É o destinatario final do producto, a quen vai dirixido. Non intervén directamente na realización do proxecto.

Cliente

Fai posíbel o proxecto. Estes usuarios deben involucrarse activamente no proxecto aportando ideas, realizando suxerencias e indicando necesidades.

4.3. Ciclo de proceso

En Scrum, cada unha das iteracións levadas a cabo polo equipo de desenvolvemento denomínase sprint. Un sprint non debe ter unha duración demasiado pequena, xa que non permitiría ao equipo realizar un producto suficientemente grande e desta maneira tamén evita que o cliente modifique os criterios moi frecuentemente. Tampouco pode ter unha duración demasiado longa pois desa maneira a desviación respecto ao que o cliente quere pode ser moi grande. O tempo habitual de duración dun sprint encóntrase entre 2 e 4 semanas. Estes tempos non están fixos, son habituais ou recomendados, polo que un sprint pode durar menos que o previo. A continuación describirase un ciclo completo de desenvolvemento dentro de Scrum.

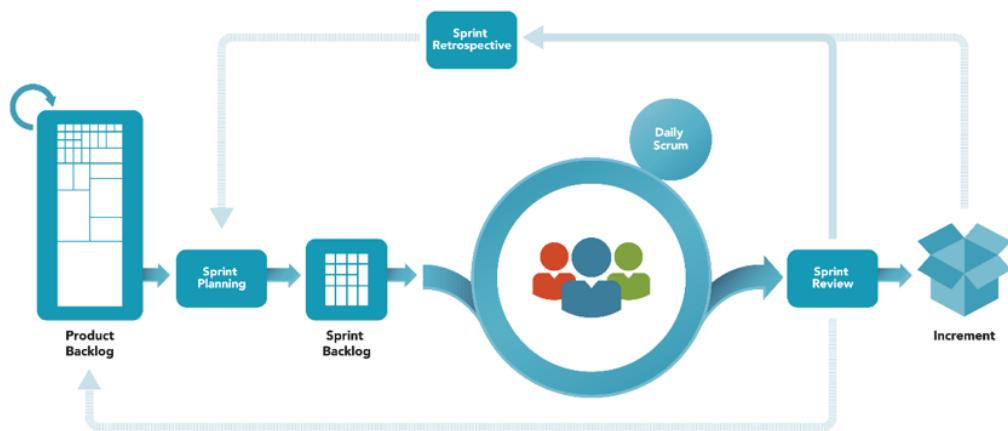


Figura 4.1: Proceso de desenvolvemento en Scrum.

4.3.1. Inicio de ciclo

O ciclo iníciase coa xuntanza dos usuarios e clientes na que se describirán as súas necesidades e aportarán suxerencias. Posteriormente, o Product owner será o encargado de crear a lista coas tarefas ordenadas por prioridade en base a esas necesidades e suxerencias dos usuarios e clientes. Esa lista é o Product backlog.

4.3.2. Sprint planning meeting

Iníciase coa xuntanza para a planificación do sprint. O Scrum master estima, coa axuda do equipo Scrum, as tarefas incluídas no Product backlog e decídese cales se levarán a cabo no sprint en base ás prioridades marcadas polo Product owner. Esta decisión tómala o Scrum master e o Product owner. Unha vez seleccionadas e priorizadas obtense o Sprint backlog, o conxunto de tarefas ordenadas para a iteración. Debe ser unha relación de tarefas realista, xa que o sprint backlog non pode ser modificado durante o sprint. Nesta xuntanza elabórase un documento chamado Sprint goal, no que se expoñen os acordos aos que se chegaron para alcanzar a seguinte entrega.

4.3.3. Daily standup

Durante o sprint fanse unhas xuntanzas diárias cunha duración máxima de quince minutos nas que se expón o estado do proxecto. Estas xuntanzas deben realizarse sempre á mesma hora e no mesmo lugar. Cada membro do equipo deber responder tres preguntas que axudan a dar unha visión do estado do proxecto:

- Que fixeches dende a última Daily standup?
- Que tes pensado facer ata a próxima Daily standup?
- Houbo algúns problemas que che impedise alcanzar o teu obxectivo?

4.3.4. Sprint review meeting

Ao rematar o sprint hai que levar a cabo unha xuntanza para a súa revisión, na que se identifica o traballo completado e o que quedou inconcluso. Nesta xuntanza tamén se

fai unha demostración do produto desenvolvido ao Product owner e ao cliente. Todas aquelas tarefas que se ensinen deben estar rematadas, non se mostra o producto a medias.

4.3.5. Sprint retrospective

Esta xuntanza tamén se realiza ao finalizar os sprints. O equipo ao completo avalia o que se fixo ben e o que se fixo mal para poder aprender dos errores e repetir os éxitos. Todos os membros do equipo teñen a posibilidade de comentar a súa opinión sobre o sprint co obxectivo de procurar unha mellora no proceso.

4.4. Vantaxes e desvantaxes de Scrum

4.4.1. Vantaxes

- Os riscos son rapidamente identificados debido ás xuntanzas diarias, polo que poden ser tratados antes de que produzan problemas.
- Axeitado para desenvolvimentos con moitos riscos xa que poden ser rapidamente codificados e probados.
- O avance no desenvolvimento do proxecto é claro debido ás xuntanzas diarias.
- O cliente pode revisar produtos frecuentemente, o que permite adaptarse a cambios suxeridos por el máis rapidamente debido ao seu feedback.
- As xuntanzas diarias permite medir a produtividade individual. Isto axuda á mellora na produtividade de cada un dos membros do equipo.
- Evita o custo asociado á xestión do proceso, o que provoca un proxecto máis barato.

4.4.2. Desvantaxes

- Requiere a plena disposición dos membros do equipo.
- Non se axusta demasiado ben a proxectos grandes con varios equipos traballando en conxunto.

- Esta metodoloxía require de membros con experiencia, xa que a planificación non depende do traballador que realice a tarefa. Tamén se debe a que todos os membros deben realizar análises, deseños, implementacións e probas.
- Se algunha das tarefas non está ben definida, o tempo e os custos do sprint non será realista. Este problema pode prolongarse durante varios sprints.
- Se a xerencia non confía completamente nos membros do equipo esta metodoloxía non é axeitada pois dá moita liberdade.
- Se non se ten unha data de fin do proxecto clara e definida o cliente pode seguir demandando novas tarefas sen límite.

4.5. Adaptación da metodoloxía

Aínda que se teñan claras todas as vantaxes propias desta metodoloxía, non se pode aplicar ao pé da letra pois o equipo de desenvolvemento consta dunha única persoa. Debido a isto, débese adaptar a metodoloxía ao noso caso. Por unha parte optouse por non levar a cabo as xuntanzas diarias e pola outra, que as reunións co titor funcionasen como as xuntanzas de inicio e fin de sprint, coincidindo no tempo. A duración dos sprints non serán fixas durante o proxecto, senón que estarán comprendida entre os 15 e os 20 días, aproximadamente. Os roles estarán repartidos entre o alumno, que será tanto o equipo de desenvolvemento coma o scrum master no día a día; e o titor, que se encargará de adoptar a figura de scrum master nas xuntanzas entre sprints.

Capítulo 5

Análise

Neste apartado levaremos a cabo a análise dos requisitos. Explicaremos os distintos actores involucrados na aplicación e os casos de uso que poderá realizar cada un deles. Debido á utilización dun método iterativo como é Scrum, a fase de análise foi realizada en cada iteración. A pesar disto, describiranse todos xuntos para simplificar o proceso, sen ter en conta se foron introducidos inicialmente ou a través dunha iteración intermedia.

5.1. Actores

No modelo de casos de uso os actores representan aos distintos usuarios que usarán o sistema e a maneira na que o farán. Dentro do noso sistema diferenciaremos tres actores distintos:

- Administrador do sistema: É o encargado de crear os edificios dentro da nosa base de datos, permitindo desa maneira que os xestores de contido dos edificios poidan realizar accións sobre el. Tamén deben crear as contas de Situm e configurar os edificios para que poidan ser utilizados co seu sistema de localización. É recomendábel que teña un perfil máis técnico que lle permita facer este tipo de accións aínda que non é unha obriga xa que cunha aprendizaxe rápida calquera persoa sería capaz de realizalas.

- Xestor do contido: Neles recae a obriga de preparar os puntos de interese e os percorridos dentro dos edificios, dende a súa localización no mapa ata os datos que estarán dispoñíbeis polo resto de usuarios. Deben ter coñecemento sobre os museos que permita a creación de percorridos en base a certos criterios artísticos, como poden ser lista de pinturas dun artista, obras contemporáneas, un mesmo estilo escultórico, etc. Un usuario Android rexistrado poderá converterse en xestor de contido sempre e cando o administrador do sistema así o indique.
- Usuario Android: É o usuario final da aplicación. Poderá situarse, localizar puntos de interese, utilizar rutas, entre outras funcionalidades proporcionadas pola aplicación. Será o actor máis utilizado posto que os usuarios que descarguen a aplicación non terán permisos especiais para realizar modificacións sobre os datos dos edificios.

5.2. Casos de uso

A continuación detallaremos os diferentes casos de uso que poderán levar a cabo os actores da aplicación. Dividirémoslos en subseccións por usuarios que poderán realizar cada un deles. Os casos de uso do Usuario Android son compartidos polo Xestor de contido.

5.2.1. Usuario Android

Neste punto describiremos os casos de uso do Usuario Android. Este é o rol que recibirán a mayoría de usuarios do noso sistema e no cal non poderán modificar a información da nosa plataforma.

Pantalla inicial

A continuación exporemos os distintos casos de uso do Usuario Android dentro da pantalla inicial. Na figura 5.1 pódense observar estas funcións.

- Autenticarse: Este caso de uso é totalmente opcional xa que se permite usar o noso sistema de maneira anónima utilizando a aplicación. O usuario pode utilizar

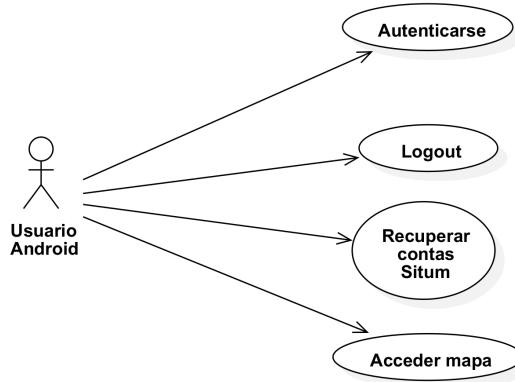


Figura 5.1: Casos de uso do Usuario Android dentro da pantalla inicial.

a súa conta de Google para identificarse dentro do noso sistema. Ao ser unha aplicación dirixida a Android non consideramos que sexa un impedimento para o uso xa que é preciso ter unha para utilizar o sistema operativo. Ver [A.1](#).

- Logout: Permitiremos a opción de facer logout na aplicación para seleccionar outra conta distinta de Google ou acceder ao sistema de maneira anónima xa que non é impedimento non estar autenticado. Ver [A.2](#).
- Recuperar contas Situm: No noso sistema permitimos o uso de varias contas de Situm, que son as que teñen permisos para localizarse dentro dos edificios. Non todas as contas de Situm estarán dispoñíbeis, poderanse restrinxir a nivel de usuario, facéndoas privadas. Debido que non é obligatoria a autenticación para poder utilizar a aplicación, debemos permitir a existencia de contas que sexan públicas para os usuarios que non se autentiquen. Ver [A.3](#).
- Acceder mapa: Unha vez se escolla a conta de Situm poderase acceder ao mapa no que se visualizarán os edificios aos que ten acceso esa conta de Situm. Ver [A.4](#).

Localización

No seguinte punto describense as funcións relacionadas coa posición do usuario e a súa visualización dentro do mapa proporcionadas por Situm. Pódense observar na figura [5.2](#).

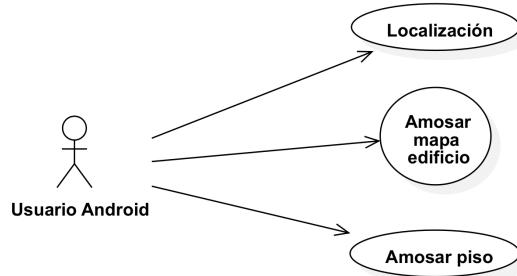


Figura 5.2: Casos de uso referentes á localización do Usuario Android.

- Amosar mapa edificio: Se a conta de Situm seleccionada ten permisos para visualizar un edificio, a aplicación permitirá amosar os seus planos incluídos na plataforma de Situm. Esta acción realizarase sobre o mapa xeral que sirva de base para a aplicación Android e nas mesmas coordenadas onde se atopa realmente o edificio, superpoñendo os planos á súa situación real. Ver [A.5](#).
- Amosar piso: Cando se teña un edificio seleccionado o cal se estea a amosar no mapa xeral, os usuarios terán a opción de cambiar o nivel que está a ser visualizado. Desta maneira poderase observar a estrutura de cada un dos pisos do edificio. Ver [A.6](#).
- Localización: Se o usuario selecciona unha conta de Situm que ten permisos sobre certo edificio e se atopa fisicamente dentro del, a aplicación identifica esa localización con gran precisión e amosa un icono indicando esa posición. Ver [A.7](#).

POIs e percorridos

No seguinte punto describense as funcións relacionadas coa visualización dos puntos de interese (POIs) dun edificio e da conexión entre eles formando percorridos. Pódense observar na figura [5.3](#).

- Amosar lista de POIs: Unha vez seleccionado un edificio, permítense a visualización dunha lista con todos os puntos de interese propios dese edificio. Ver [A.8](#).

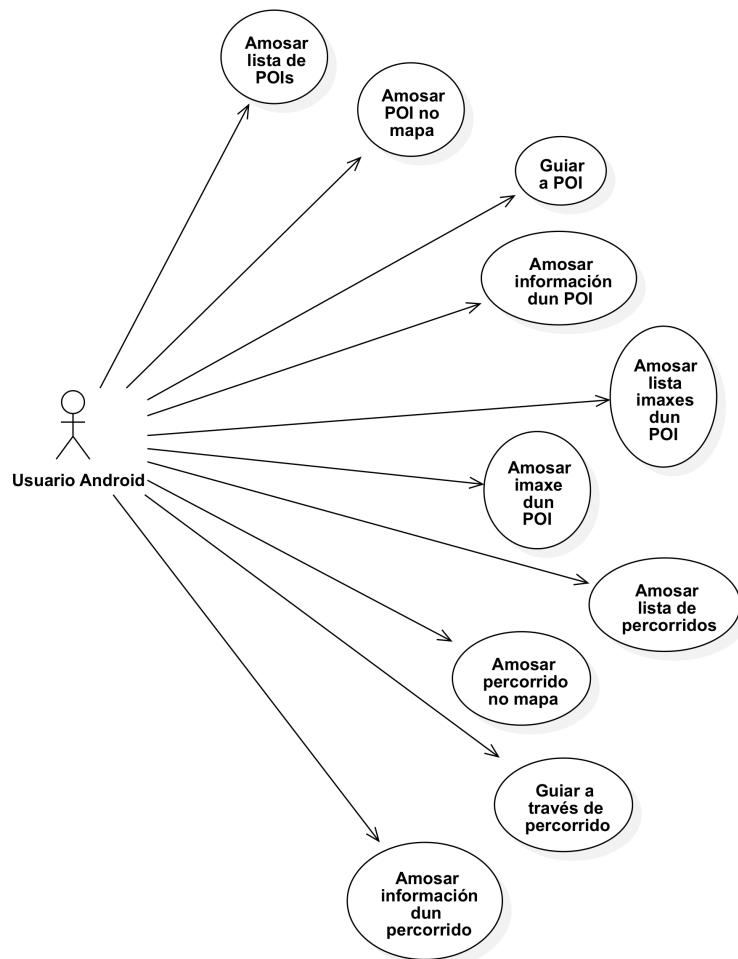


Figura 5.3: Casos de uso referentes á visualización de POIs e percorridos.

- Amosar POI no mapa: Seleccionando un POI da lista de puntos de interese situarse un marcador no mapa indicando a posición exacta dese punto. Ver [A.9](#).
- Guiar a POI: Habilítase a opción de guiado a calquera punto de interese existente no edificio a través das opcións dadas por Situm. As ordes proporcionadas permitirán avanzar pola ruta más curta ata o punto desexado grazas á configuración de Situm. Ver [A.11](#).
- Amosar información dun POI: Permite ao usuario visualizar todos os datos relativos a un punto de interese dun edificio gardada no noso sistema. Ver [A.10](#).

- Amosar lista de imaxes dun POI: Unha vez seleccionado un punto de interese, permítese a visualización dunha lista con todas as imaxes dese POI. Ver [A.16](#).
- Amosar imaxe dun POI: De entre a lista de imaxes dun POI permitirase a selección dunha imaxe concreta para visualizala na pantalla. Ver [A.17](#).
- Amosar lista de percorridos: Unha vez seleccionado un edificio, permítese a visualización dunha lista con todos os percorridos propios dese edificio. Ver [A.12](#).
- Amosar percorrido no mapa: Seleccionando un percorrido da lista de percorridos situárase un marcador no mapa por cada punto de interese que compoña ese percorrido, uníndoos mediante liñas que mostren a dirección na cal se ten que realizar. Ver [A.13](#).
- Guiar a través dun percorrido: Habilítase a opción de guiado a través de calquera percorrido existente no edificio a grazas ás opcións dadas por Situm. Baséase no caso de uso "Guiar a POI", pero visualizarse no mapa o percorrido xa realizado e o que aínda non se realizou.. Ver [A.14](#).
- Amosar información dun percorrido: Permite ao usuario visualizar toda a información relativa a un percorrido dun edificio gardada no noso sistema. Ver [A.15](#).
- Actualizar ruta: Permite ao usuario visualizar toda a información relativa a un percorrido dun edificio gardada no noso sistema. Ver [A.15](#).

5.2.2. Xestor de contido

A continuación exporemos os casos de uso do Xestor de contido. Para a realización destes casos de uso é preciso ter permiso de edición sobre o edificio en cuestión. A parte dos seus propios casos de uso tamén pode realizar todas as accións do Usuario Android.

Puntos de interese

No seguinte punto describense as funcións relacionadas coa creación e edición dos puntos de interese (POIs) dun edificio. Pódense observar na figura [5.4](#).

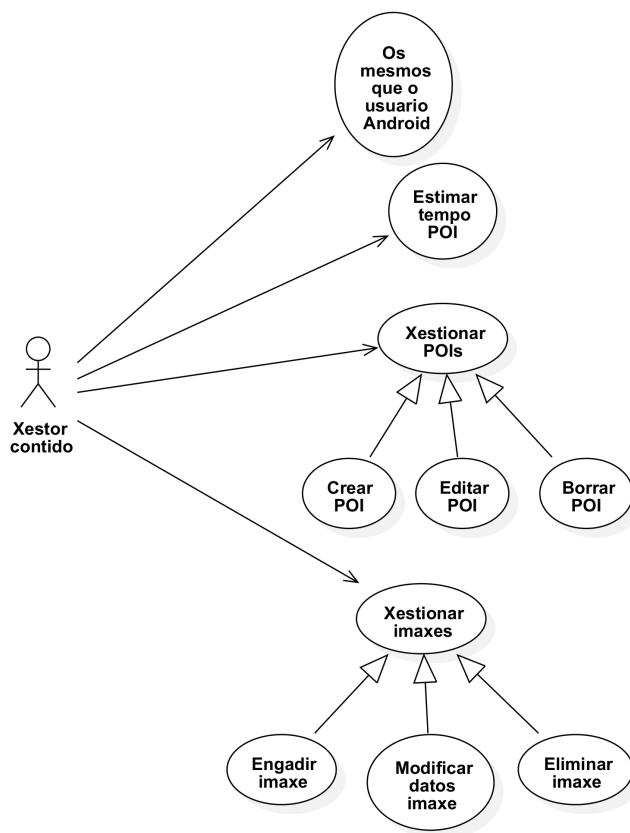


Figura 5.4: Casos de uso do Xestor de contenido referentes aos POIs.

- Xestión de POIs - Creación: Unha vez seleccionado un edificio, permítese a creación dun punto de interese dentro dos diversos niveis nos que se pode dividir o edificio. Débese seleccionar a posición do novo punto e darlle un nome e unha descripción. Ver [A.18](#).
- Xestión de POIs - Edición: Seleccionando un POI da lista de puntos de interese poderase entrar nunha pantalla onde visualizar toda a súa información. Dende ese punto permítese a edición dos seus datos. Ver [A.19](#).
- Xestión de POIs - Borrado: Dende a mesma pantalla onde se visualiza e modifica a información dun POI débese permitir a eliminación dese punto, sempre que non estea incluído dentro dalgún percorrido. Ver [A.20](#).

- Estimar tempo POI: Dende a mesma pantalla onde se visualiza e modifica a información dun POI débese permitir a estimación do tempo adicado a ese punto.

Ver [A.21](#).

- Xestión de imaxes - Engadir: Existe a opción de incluír e asociar imaxes a distintos puntos de interese que permitan describir e completar a información dada sobre eles. Tamén será dende a pantalla de visualización dos datos dun POI onde se permitirá engadir imaxes, aportando información sobre elas. Ver [A.22](#).
- Xestión de imaxes - Modificar datos: Unha vez seleccionada unha imaxe dun punto de interese, permítese a modificación dos datos desa imaxe. Ver [A.23](#).
- Xestión de imaxes - Eliminar: De entre a lista de imaxes dun POI permitirase a selección dunha imaxe concreta para eliminala do noso sistema. Ver [A.24](#).

Percorridos

Na figura [5.5](#) pódense observar todas as funcións que pode levar a cabo o actor relacionadas cos percorridos.

- Xestión de percorridos - Creación: Permitiremos a creación de percorridos dende a pantalla principal, onde seleccionaremos todos os puntos de interese que o compoñan, paso previo á introdución da información asociada a ese percorrido dende a pantalla de inserción de datos. Non se poden repetir puntos de interese dentro do mesmo percorrido. O número mínimo será de 3 POIs. Ver [A.25](#).
- Xestión de percorridos - Modificar información: Unha vez seleccionado un percorrido dentro dun edificio, permítese a modificación dos seus datos. Ver [A.26](#).
- Xestión de percorridos - Eliminación: Unha vez seleccionado un percorrido dun edificio, permítese a eliminación do mesmo dende a pantalla de modificación da súa información. Ver [A.27](#).
- Estimar tempo percorrido: Dende a mesma pantalla onde se visualiza e modifica a información dun percorrido débese permitir a estimación do tempo adicado a ese punto. Ver [A.28](#).

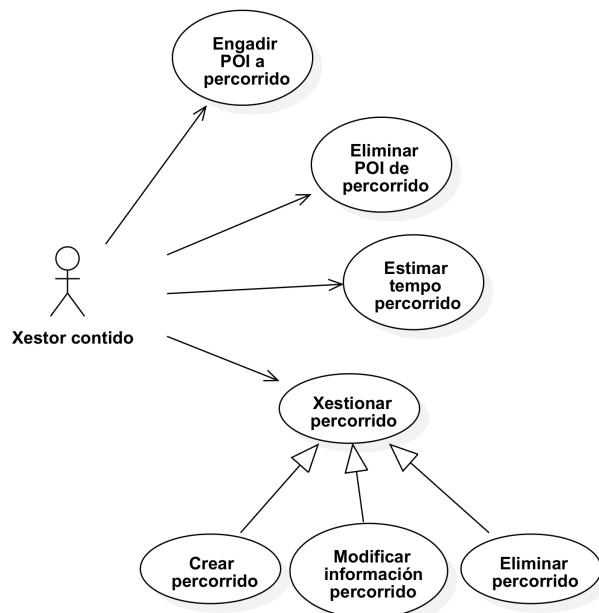


Figura 5.5: Casos de uso do Xestor de contenido referentes aos percorridos.

- Engadir POI a percorrido: Despois de seleccionar un percorrido disporase da posibilidade de engadir puntos de interese non incluídos xa dentro do mesmo. Esta inserción poderá ser ao inicio do percorrido, no final, ou tamén entre dous POIs xa incluídos. Ver [A.29](#).
- Eliminar POI de percorrido: Poderase escoller entre todos os POIs incluídos nun percorrido para a súa eliminación, sempre que non se deixe ao percorrido con menos de 3 POIs despois da eliminación. Ver [A.30](#).

5.2.3. Administrador do sistema

A continuación exporemos os casos de uso do Administrador do sistema. As accións levadas a cabo por este actor non se realizarán dende a aplicación, senón que haberá que utilizar algúun xestor de base de datos. Na figura 5.6 pódense observar estas accións.

- Dar de alta un edificio: Este será o primeiro paso para a introdución dun novo edificio dentro da nosa plataforma. Antes de poder engadir POIs e percorridos de-

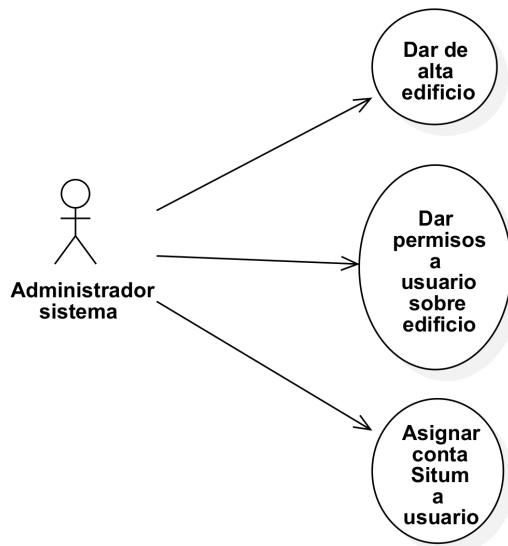


Figura 5.6: Casos de uso do Xestor de contenido referentes aos percorridos.

bemos inserir o edificio na base de datos, incluíndo información propia do sistema de Situm que nos permita ligar ambas plataformas. Ver [A.31](#).

- Dar permisos a un usuario sobre algún edificio: Para permitir a creación de contenido sobre os edificios débense dar permisos de xestor a algún usuario. Estes usuarios deben existir na base de datos, é dicir, debéronse autenticar nalgún momento na aplicación para poder ter os seus datos. Os permisos serán outorgados individualmente sobre cada edificio, para que un usuario non poida modificar todos os edificios dunha conta. Ver [A.32](#).
- Asignar contas Situm a usuarios: Como non todos as contas de acceso ao sistema de Situm van estar visíbeis para calquera usuario que entre na aplicación, debemos asignar esas contas privadas a certos usuarios. Desta maneira, cando algún usuario acceda á aplicación, poderá ver tanto as contas de Situm públicas coma as privadas que estean asociadas ao seu usuario. Ver [A.34](#).

Capítulo 6

Planificación e custos

6.1. Desenvolvemento

O desenvolvemento desta aplicación levouse a cabo en varios sprints ou iteracións Scrum, creando para cada un deles o Sprint backlog, é dicir, o conxunto de funcionalidades a implementar. Os casos de uso desenvolvidos fóreronse incrementando, aumentando dese xeito o número de funcionalidades dispoñíbeis.

Cada una destas iteracións foi dividida en catro fases distintas que se identifican coa planificación clásica dun proxecto: análise, deseño, implementación e probas.

No inicio do desenvolvemento decidiuse que a duración de cada sprint sería de 30 días.

Antes de comezar o primeiro sprint adicouse tempo á formación na programación para Android, así coma no uso de Android Studio e a realización de probas nun terminal físico e emulado pois era a primeira vez que se utilizaban estas tecnoloxías.

6.1.1. Sprint 1: Aplicación Android: localización dentro dun edificio

O obxectivo desta primeira iteración é a creación dos elementos básicos para a utilización dos sistemas de localización en interiores de Situm, permitindo aos usuarios ter un primeiro achegamento a esta tecnoloxía. Nesta primeira iteración só precisaremos construír unha aplicación para Android, deixando funcionalidades que requiran máis infraestrutura para seguintes sprints.

Para este sprint non se implementará a selección de contas de Situm nin a autenticación, polo que o rol dos usuarios que a utilicen será de Usuario Android anónimo cunha conta de Situm fixa. Só existirá unha actividade cun fragmento onde se mostrará o mapa.

A continuación veremos unha lista cos casos de uso implementados nesta iteración:

- Visualizar o mapa dun edificio. Ver caso de uso [A.5](#).
- Localizar un usuario dentro dun edificio. Ver caso de uso [A.7](#).
- Cambiar de piso dentro dun edificio. Ver caso de uso [A.6](#).

Fito Ao final deste sprint teremos unha aplicación simple pero funcional, na que se poderá localizar usuarios dentro de edificios configurados para a súa utilización con Situm. Permitirá ao cliente observar as posibilidades de localización de Situm dentro dun edificio e comprobará a súa exactitude.

Proba A proba para validar as tarefas implementadas será a utilización da aplicación dentro dun edificio con varios niveis, movéndose en toda a súa extensión. Desta maneira observaranse as posibilidades de localización da nosa plataforma e o bo funcionamento do cambio de niveis, tanto fisicamente como seleccionándoo no mapa.

6.1.2. Sprint 2: Servidor inicial: POIs e percorridos

O obxectivo desta segunda iteración é a creación das funcións básicas do servidor e o seu despregue para poder ser utilizado nun entorno de producción. Neste punto aínda non se traballará na integración coa aplicación Android senón que nos centraremos únicamente no servidor.

Traballarase na arquitectura xeral do servidor e nas súas diferentes capas, polo que ao final deste sprint teremos uns servizos web ampliábeis de xeito doado. Crearanse servizos para a recuperación de información sobre os puntos de interese e os percorridos almacenados nunha base de datos.

O administrador do sistema será o encargado de inserir a través dun programa xestor os datos recuperados pola aplicación. Estes datos serán novos edificios, puntos de interese e percorridos.

A continuación veremos os casos de uso da parte servidora nos que se traballou neste sprint:

- Dar de alta edificio. Ver caso de uso [A.31](#).
- Amosar lista de POIs. Ver caso de uso [A.8](#).
- Amosar información dun POI. Ver caso de uso [A.10](#).
- Amosar lista de percorridos. Ver caso de uso [A.12](#).
- Amosar información dun percorrido. Ver caso de uso [A.15](#).

Fito Ao final deste sprint obteremos un servidor utilizábel dende calquera lugar e cuns casos de uso básicos para comprobar o seu funcionamento.

Proba As probas realizadas serán levadas a cabo nun entorno local e noutro de producción, co despregue do servidor a través dos servizos de Amazon (AWS). Realizaranse chamadas para recuperar a información a través dos servizos publicados.

6.1.3. Sprint 3: Lectura datos dende aplicación Android

Na terceira iteración do noso proxecto búscase a ligazón entre a aplicación Android e o servidor, que ata este punto levaron camiños paralelos. Modificarase a aplicación Android para recibir a información fornecida polo servidor, procesarla e mostrala por pantalla. Neste sprint tampouco se traballará con outro rol que non sexa o de Usuario Android anónimo.

En canto aos casos de uso tratados nesta iteración, aparte de completar coa parte de vista os comezados no sprint anterior, implementáronse os seguintes:

- Amosar POI no mapa. Ver caso de uso [A.9](#).
- Amosar percorrido no mapa. Ver caso de uso [A.13](#).

Fito Ao final deste sprint teremos por primeira vez unha visión xeral do que proporcionará a nosa plataforma, unindo a potencia de Situm cos nosos propios datos, permitindo amosar puntos de interese e percorridos. Poderase visualizar no mapa da aplicación os POIs e percorridos almacenados na nosa base de datos e acceder á súa información.

Proba A proba para validar as tarefas implementadas consistirá na utilización da aplicación nun edificio configurado en Situm con información sobre puntos de interese e percorridos no noso sistema.

6.1.4. Sprint 4: Autenticación de usuarios

Neste sprint tratarase a autenticación dos usuarios a través dos servizos de Google. Almacenaremos estes usuarios autenticados para establecer permisos nos distintos edificios. Tamén se permitirá a selección de contas de Situm, polo que debemos proporcionar unha lista coas contas dispoñíbeis sexan públicas ou estean asociadas ao usuario autenticado.

A continuación veremos unha relación cos casos de uso nos que se traballou neste sprint:

- Autenticarse a través de Google. Ver caso de uso [A.1](#).
- Facer logout. Ver caso de uso [A.2](#).
- Dar de alta unha conta de Situm. Ver caso de uso [A.33](#).
- Recuperar as contas de Situm dispoñíbeis. Ver caso de uso [A.3](#).

Fito Ao final deste sprint teremos unha actividade inicial onde o usuario poderá autenticarse a través dunha conta de Google para acceder á aplicación. Esta autenticación tamén lle proporcionará a posibilidade de ter acceso a máis contas de Situm, o que provocará que poida ver novos edificios asociados a esas contas.

Proba A proba para validas este sprint consistirá na utilización da aplicación con varias contas de Google que teñan distintos permisos sobre contas de Situm, para comprobar desta maneira que non están todas dispoñíbeis para usuarios anónimos ou con distintas contas de Google.

6.1.5. Sprint 5: Edición de POIs e percorridos dende a aplicación Android

Ata este sprint a creación dos datos para a nosa aplicación tiña que ser directamente sobre a base de datos cun xestor. Neste sprint cubriranse os casos de uso que permitan realizar estas modificacíons directamente dende a aplicación Android, incorrendo nun menor esforzo. Neste sprint fai a súa aparición o rol Xestor de contido, que será o encargado da xestión dos puntos de interese e percorridos.

A continuación veremos unha relación cos casos de uso nos que se traballou neste sprint:

- Xestión de POIs: Creación. Ver caso de uso [A.18](#).
- Xestión de POIs: Modificación. Ver caso de uso [A.19](#).
- Xestión de POIs: Borrado. Ver caso de uso [A.20](#).
- Xestión de percorridos: Creación. Ver caso de uso [A.25](#).
- Xestión de percorridos: Modificación. Ver caso de uso [A.26](#).
- Xestión de percorridos: Borrado. Ver caso de uso [A.27](#).
- Engadir POI a percorrido. Ver caso de uso [A.29](#).
- Eliminar POI de percorrido. Ver caso de uso [A.30](#).
- Dar permisos a usuario sobre un edificio. Ver caso de uso [A.32](#).

Fito Ao final deste sprint teremos unha aplicación completamente funcional, coa posibilidade da edición dentro da mesma sen ter que recorrer a elementos externos.

Proba O primeiro paso para as probas deste sprint consistirá na creación, modificación e borrado de POIs ao longo dun edificio, en distintas plantas e con diferentes usuarios. Posteriormente utilizaremos eses puntos de interese para a creación de diversos percorridos, que tamén serán editados (tanto a súa información coma a inserción de POIs despois de ter creado o percorrido) e borrados. Tamén se deberá comprobar as opcións dispoñíbeis para os usuarios en base aos roles asignados.

6.1.6. Sprint 6: Valor engadido da localización Situm

O obxectivo do sexto sprint é a utilización da localización en Situm para proporcionarlle máis servizos ao usuario. Non se precisará ningún permiso especial para utilizar esta capacidade, que permitirá guiar aos usuarios a un punto de interese ou seguir todo un percorrido. A ruta de guiado será actualizada automaticamente pola aplicación mentres o usuario non o cancele. Tamén se lle permitirá aos xestores de contido estimar o tempo adicado a cada POI e percorrido.

A continuación veremos unha relación cos casos de uso nos que se traballou neste sprint:

- Guiar ata un punto de interese. Ver caso de uso [A.11](#).
- Guiar a través dun percorrido. Ver caso de uso [A.14](#).
- Estimar tempo adicado a POI. Ver caso de uso [A.21](#).
- Estimar tempo adicado a percorrido. Ver caso de uso [A.28](#).

Fito Ao final deste sprint o usuario poderá ser guiado aos puntos de interese do edificio, así como poder utilizar esta capacidade en edificios, sabendo canto tempo estimado lle levará cada visita grazas aos novos valores de cada POI e percorrido.

Proba As probas para validar este sprint consistirán no guiado dentro dun edificio configurado a través de Situm mediante as rutas creadas no seu dashboard.

6.1.7. Sprint 7: Imaxes para POIs

O obxectivo do séptimo sprint é a xestión de imaxes para os puntos de interese. Os xestores de contido poderán engadir as imaxes que consideren oportunas a calquera POI dentro dun edificio para que visualicen os usuarios da aplicación. Non se precisará ningún permiso a maiores, chega con ser xestor de contido dun edificio. Non haberá límite de imaxes para os puntos.

A continuación veremos unha relación cos casos de uso nos que se traballou neste sprint:

- Amosar lista de imaxes dun POI. Ver caso de uso [A.18](#).
- Amosar imaxe dun POI. Ver caso de uso [A.19](#).
- Xestión de imaxes: Engadir. Ver caso de uso [A.25](#).
- Xestión de imaxes: Modificar datos. Ver caso de uso [A.26](#).
- Xestión de imaxes: Eliminar. Ver caso de uso [A.27](#).

Fito Ao final deste sprint teremos a opción de administrar imaxes sobre os puntos de interese e visualizalas para permitir dar maior información sobre cada un deles.

Proba As probas para validar este sprint consistirán no engadido e eliminación de imaxes dentro dos POIs dun edificio por parte dun xestor de contido. Débese comprobar que a un usuario Android normal non se lle permita o engadido ou eliminación de imaxes. Ambos roles (xestor de contido e usuario Android) deben poder ver as imaxes dos POIs.

6.2. Planificación e custos iniciais

A continuación pódense observar tanto a planificación inicial do proxecto coma os custos asociados a esta. No terceiro punto descríbense os plans de continxencia preparados para o proxecto.

6.2.1. Planificación inicial

Nesta sección describirase a planificación completa e detallada do proxecto. O primeiro paso para realizala foi establecer o alcance do proxecto e os requisitos precisos para poder levalo a cabo. Na primeira planificación fíxose un bosquexo xeral dos pasos a seguir no proxecto, estimando unha duración do desenvolvemento contando con sete sprints dunhas tres semanas cada un. Debido ao horario dispoñíbel polo traballador, considerouse unha xornada laboral de catro horas diárias.

Pódese observar o diagrama de Gantt coa planificación detallada na figura 6.1. Diferéncianse todos os pasos realizados no proxecto, así como o tempo adicado a cada sprint.

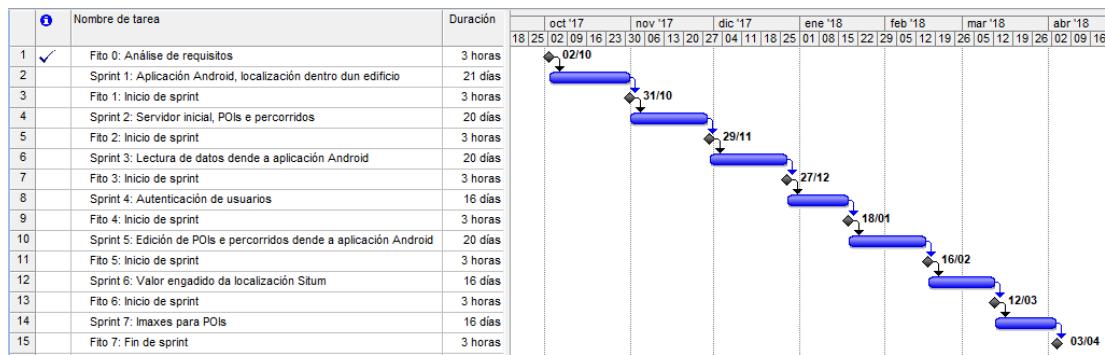


Figura 6.1: Diagrama de Gantt da planificación inicial para os sprints.

6.2.2. Custo inicial

Unha vez realizada a planificación das tarefas nas que se divide o proxecto pódese comenzar coa análise económica dos custos asociados. Como é evidente, o custo máis alto recae sobre os recursos humanos, xa que os materiais utilizados non demandaron moita inversión.

Recursos humanos Ao ser un proxecto elaborado por un único traballador, tivo que realizar varias funcións que normalmente recaerían sobre distintas persoas, polo que asignamos distintos valores ao seu traballo dependendo da función que realizase en

cada momento. Considérase xefe de proxecto ao titor do proxecto que realiza tarefas de supervisión e exerce de scrum master nas xuntanzas entre sprints. O tempo estimado para o xefe de proxecto coincide cos fitos de inicio e fin de sprint, polo que terá que actuar 8 veces. Na táboa 6.1 pódese observar a descomposición da estimación inicial por tarefa para o analista e o programador.

Sprint	Analista	Programador
Primeiro	14 h	70 h
Segundo	11 h	69 h
Terceiro	13 h	67 h
Cuarto	10 h	54 h
Quinto	13 h	67 h
Sexto	10 h	54 h
Sétimo	10 h	54 h

Táboa 6.1: Horas planificadas para o traballador.

Pódense observar os custos asociados aos recursos na táboa 6.2.

Recurso	Salario (€/h)	Tempo (h)	Custo
Xefe de proxecto	50	24	1200 €
Analista	37	81	2997 €
Programador	20	516	10320 €
Formación	0	40	0 €
Total			14517 €

Táboa 6.2: Custos planificados en recursos humanos.

Recursos materiais Podemos dividir os recursos materiais en hardware e software. Non houbo gastos referidos ao software posto que se utilizou software gratuito na realización do proxecto, mais si que houbo gastos en canto aos recursos hardware. Houbo que realizar un desembolso polo ordenador utilizado para a programación do proxecto posto que a emulación dun terminal Android é bastante custosa e unha computadora antiga non era capaz de executalo. Tamén se inclúen nos gastos un móvil de gama media para a proba da aplicación en entornos reais. Mención aparte require a subscrpción a Amazon Web Services, posto que se desfrutou dun período de proba que, non obstan-

te, non estaría dispoñíbel unha vez a aplicación requira un maior uso de recursos por parte do servidor. Este tipo de servizos supoñen un maior custo canta máis demanda procesan, polo que en entornos reais de producción o custo aumentaría. Os gastos debido aos recursos materiais pódense ver na táboa 6.3. Unicamente imputamos ao proxecto o custo do PC de sobremesa posto que o teléfono móvil foi utilizado para outros usos á marxe do proxecto e polo tanto, amortizado.

Recurso	Custo	Imputado ao proxecto
PC sobremesa	860 €	860 €
Teléfono móvil	440 €	0 €
Total		860 €

Táboa 6.3: Custos en recursos materiais.

Total Despois de analizar os custos en recursos humanos e materiais podemos establecer o custo total inicial. Obsérvanse todas as entradas na táboa 6.4:

Tipo de recurso	Custo
Recursos humanos	14517 €
Recursos materiais	860 €
Total	15377 €

Táboa 6.4: Custos totais estimados.

6.2.3. Análise de riscos e plans de continxencia

Neste punto trátase a análise de todos os riscos más importantes que se poden producir na elaboración do proxecto xunto coa posibilidade de que se produzan, para despois elaborar plans de continxencia para cada un deles no caso de que afectasen ao traballo.

Riscos Na táboa 6.5 pódense ver os riscos propios do proxecto realizado. Tivéronse en conta tanto a probabilidade de que se producisen estes errores (alta, media, baixa) coma o impacto sobre o desenvolvemento (alto, medio, baixo) para calcular a exposición ao

risco (alta, media, baixa) e dese xeito poder ordenalos para atacar aqueles cunha maior exposición. A continuación enumeramos os riscos detectados ao inicio do proxecto:

Risco	Probabilidade	Impacto	Exposición
Programación en Android	Alta	Baixo	Baixa
Ferramentas descoñecidas	Alta	Baixo	Baixa
Modificación de requisitos	Alta	Medio	Alta
Interface de usuario non clara	Media	Medio	Media
Dispoñibilidade do equipo	Media	Alta	Alta

Táboa 6.5: Avaliación dos riscos do proxecto.

- Programación en Android: Debido a que o equipo de desenvolvemento nunca programara ningunha aplicación para Android, había un risco importante de retraso nos primeiros sprints.
- Ferramentas descoñecidas: Ao igual que o punto anterior, o equipo de desenvolvemento non tiña experiencia en programación coa SDK de Situm, nin con Google Maps.
- Modificación de requisitos: Ao ser un proxecto con tecnoloxía novedosa e en constante actualización é complicado definir claramente e con tempo as especificacións desexadas.
- Interface de usuario non clara: En relación co primeiro punto tratado, ao ser a primeira vez que o equipo realiza unha aplicación móvil, resulta moi probábel que a interface creada para a aplicación non sexa o suficientemente clara e atractiva por falta de experiencia real.
- Dispoñibilidade do equipo de desenvolvemento: O equipo de desenvolvemento está constituído por un único membro cun traballo a xornada completa, polo que a súa dispoñibilidade non será a desexada durante a duración do proxecto.

Plans de continxencia Débese ter especial coidado cos riscos cunha alta exposición, polo que se serán os que máis atención reciban:

- Modificación de requisitos: Para evitar este risco a mellor maneira é a comunicación continua co cliente, polo que se deben ter xuntanzas periódicas co mesmo. Un dos puntos fuertes da metodoloxía Scrum é precisamente a protección contra os cambios de requisitos.
- Dispoñibilidade do equipo de desenvolvemento: Ante a imposibilidade de realizar o traballo en horario laboral, deberase reservar as fins de semana para tratar de ter un calendario estable para a creación do proxecto. Ao final de cada sprint realizarase un seguimento do esforzo realizado e formularase unha replanificación sempre que se considere necesario.
- Interface de usuario non clara: Buscarase unha selección de iconas o suficientemente representativas das accións dispoñíbeis na aplicación que permitan unha aproximación sinxela aos usuarios. Ao finalizar os sprints é recomendábel amosar as interfaces a posíbeis usuarios.
- Programación en Android: Os primeiros sprints deben ser o suficientemente folgados para unha aprendizaxe correcta das características propias da programación en Android.
- Ferramentas descoñecidas: Ao igual que ocorría no punto anterior, hai que ter en conta a programación coa SDK de Situm así coma dos métodos propios de Google Maps para que non provoquen un retraso importante.

6.3. Planificación e custos finais

Unha vez finalizado o proxecto pódese analizar a planificación final, con todas as incidencias producidas; así coma visualizar os custos reais e totais derivados do desenvolvemento.

6.3.1. Planificación final

Despois da realización do proxecto e superados todos os problemas xurdidos durante a súa elaboración, pásase a detallar a planificación final. Pódese observar o diagrama

de Gantt final detallado na figura 6.2. En total, aumentouse o tempo de proxecto en 11 días laborais. Unha vez finalizado o séptimo e último sprint do desenvolvemento, procedeuse á realización da memoria que levou tres semanas máis de traballo grazas a que durante as iteracións foise cubrindo parte da documentación. Estas últimas semanas débense á creación de parte dos apéndices así como as seccións de planificación, probas e conclusóns.

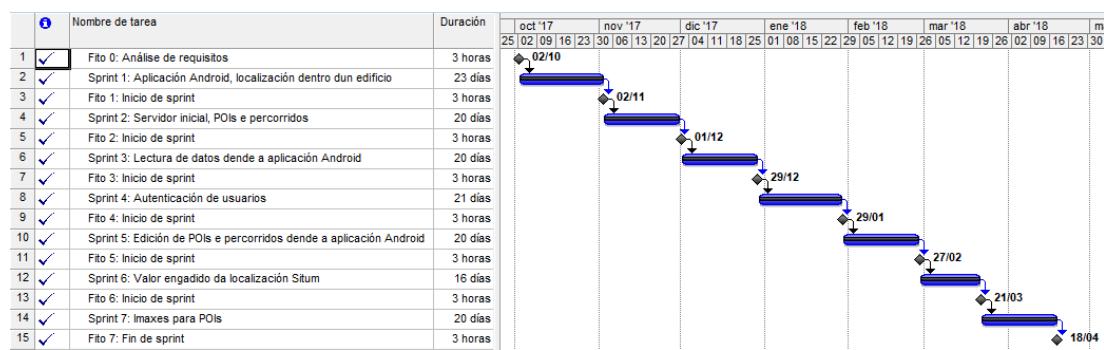


Figura 6.2: Diagrama de Gantt para os sprints despois de realizar o seguemento ao proxecto.

6.3.2. Incidencias e resolución

A continuación expóñense os problemas ocorridos durante o proxecto a como se resolvieron, divididos por sprints:

- Sprint 1: A pesar de telo presente na planificación, houbo que incrementar levemente o tempo adicado a esta iteración por mor da pouca destreza do programador nun entorno novo para el como é Android e tamén pola integración da aplicación con Google Maps. A representación dos mapas de interiores proporcionados por Situm foi outro dos puntos que afectou negativamente á planificación, mentres que o tratamiento da posición e a posta en marcha da SDK de Situm foi bastante sinxela polo que contrarrestou o efecto da visualización dos mapas.
- Sprint 2: Esta iteración levouse a cabo no tempo estimado posto que o programador xa estaba afeito a este tipo de traballo.

- Sprint 3: A pesar do crítico deste paso, non se rexistrou unha demora na súa realización.
- Sprint 4: Con esta iteración non houbo ningún problema ata o momento de publicar a aplicación na tenda de Google Play, xa que houbo que reconfigurar Google Play Services por culpa do cifrado da aplicación. Non foi ata esta reconfiguración cando se puido realizar correctamente unha autenticación na aplicación descargada dende Google Play. Ao inicio descoñecíase o motivo polo cal non se permitía esta acción, o que provocou un aumento considerábel da duración do sprint.
- Sprint 5: A pesar de ser a iteración con máis contido dentro da planificación, non supuxo un problema a súa implementación posto que xa se avanzara o suficiente no proxecto como para estar cómodo no novo entorno e os casos de uso realizados non eran especialmente complexos.
- Sprint 6: Non houbo problemas inesperados nesta iteración.
- Sprint 7: O envío e recepción de imaxes a través do dispositivo móvil foi máis problemático do inicialmente esperado. A parte do servidor foi segundo o estimado xa que se puido validar a través da ferramenta Postman mais non así a integración coa aplicación móvil.

6.3.3. Custo final

Unha vez finalizado o proxecto, débese calcular o custo final. A única variación co custo calculado ao inicio recae sobre os recursos humanos, xa que non houbo ningún tipo de variación de prezo nos recursos hardware ou software, polo que soamente se tratará a variación por culpa dos retrasos nos sprints.

Recursos humanos A continuación indicaranse as horas de aumento con respecto á planificación tanto para o analista como para o programador. Os sprints que sufrieron retrasos foron o primeiro, o cuarto e o séptimo. Na táboa 6.6 pódense ver as horas concretas.

Pódense observar o custo final en recursos humanos na táboa 6.7.

Sprint	Analista	Variación analista	Programador	Variación programador
Primeiro	14 h	0 h	78 h	8 h
Segundo	11 h	0 h	69 h	0 h
Terceiro	13 h	0 h	67 h	0 h
Cuarto	15 h	5 h	69 h	15 h
Quinto	13 h	0 h	67 h	0 h
Sexto	10 h	0 h	54 h	0 h
Séptimo	12 h	2 h	68 h	14 h

Táboa 6.6: Horas reais para o traballador.

Recurso	Salario (€/h)	Tempo estimado (h)	Tempo real	Custo
Xefe de proxecto	50	24	24	1200 €
Analista	37	81	88	3256 €
Programador	20	516	560	11200 €
Formación	0	40	40	0 €
Total				15656 €

Táboa 6.7: Custos finais en recursos humanos.

Custos totais Finalmente, súmanse os custos en recursos humanos calculados despois da finalización do desenvolvemento cos custos en recurso materiais que non se viron modificados. Na táboa 6.8 pódese ver o cálculo do custo total do proxecto unha vez realizado.

Tipo de recurso	Custo
Recursos humanos	15656 €
Recursos materiais	860 €
Total	16516 €

Táboa 6.8: Custos totais unha vez finalizado o proxecto.

Capítulo 7

Deseño

Neste capítulo explícase a arquitectura da plataforma comenzando por unha visión xeral da mesma para despois entrar en detalle no modelo de datos, o deseño do servidor, a aplicación Android e a autenticación a través de Google.

7.1. Arquitectura xeral

A nosa plataforma consta dos seguintes elementos conectados entre si:

- Servidor de base de datos (POIs e percorridos).
- Aplicación Android para visualización e edición de datos.
- Sistema de autenticación (GoogleAuth).
- Sistema de localización en interiores (Satum).

A aplicación Android precisa o servidor para a recuperación dos datos propios da plataforma e o sistema de localización en interiores para poder funcionar. O sistema de autenticación de Google está relacionado coa aplicación Android, onde comeza ese proceso, e co servidor, que realiza o último paso. Pódense observar estas conexiós entre os elementos na figura 7.1):

Servidor O servidor encárgase de recuperar e procesar a información que precisa a aplicación para o seu funcionamento. Esta información é almacenada nunha base de

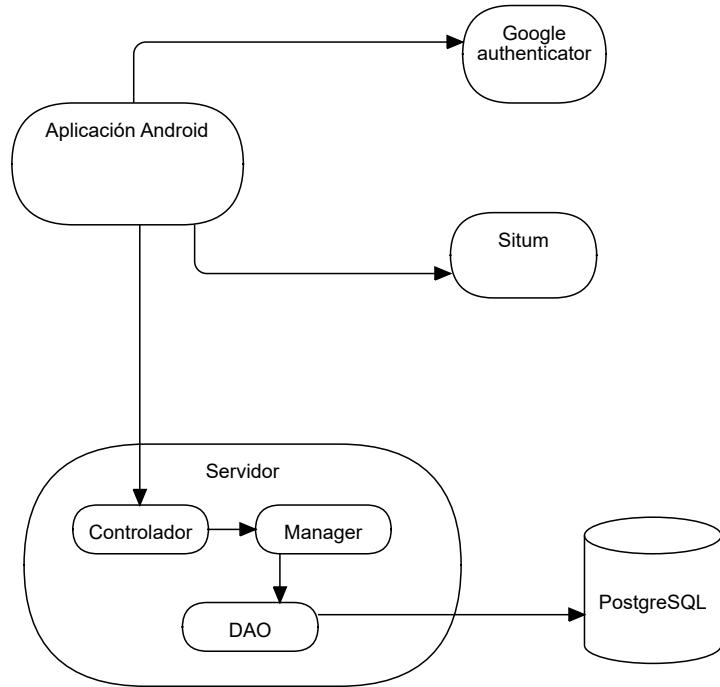


Figura 7.1: Arquitectura xeral da plataforma Caronte para a guía de museos.

datos PostgreSQL. Aparte de recuperar esta información, tamén se encarga de inserir os datos indicados polo usuario dentro da aplicación. Para permitir a comunicación coa aplicación Android e que o usuario poida recibir e modificar esa información, publícanse uns servizos web a través dunha API REST. Decidiuse despregalo en Amazon Web Services por varios motivos, entre os cales se encontran:

- Facilidade de configuración e uso, tanto para obter o propio servidor como para engadir novos elementos como pode ser a base de datos PostgreSQL.
- Escalabilidade.
- Rentabilidade.
- Seguridade.

Aplicación Android É o punto de interacción do usuario coa nosa plataforma. Presenta a información recuperada do servidor ao usuario e permite a súa modificación,

enviando de novo os datos para a súa persistencia. Tamén é a encargada de tratar os servizos de Situm que permiten a localización e o guiado en interiores, recuperando a información necesaria e presentándolla ao usuario.

Autenticación con Google O sistema de autenticación de Google permite a recuperación de información propia do usuario a través dunha conta de Google. Ao utilizar este sistema, pódese dispoñer da información dunha conta de usuario e gardala no noso servidor, polo que se pode utilizar para asociar permisos ou gardar outra información relacionada.

7.2. Modelo de datos

Neste punto exporase o modelo de datos da plataforma que se seguiu á hora da creación da base de datos. Na figura 7.2 pódese observar o diagrama entidade relación. Dividirase a explicación en base ás entidades.

7.2.1. EDIFICIO

Nesta entidade gardarase a información básica dos edificios que se tratan na aplicación. Aparte dos datos propios creados para a plataforma tamén se garda o identificador do edificio dentro do sistema Situm, que permite relacionar os edificios de Caronte cos de Situm.

As columnas das que se compón este entidade son:

- **ID_EDIFICIO:** Chave primaria autoxerada da táboa. De tipo SMALLINT (short). É o identificador do edificio dentro do sistema de Caronte. Non pode ter valor nulo.
- **ID_EDIFICIO_EXTERNO:** De tipo SMALLINT (short). É o identificador do edificio dentro do sistema de Situm.
- **NOME:** De tipo VARCHAR (string) con límite de 50 caracteres. É o texto que se visualizará na aplicación e que permite ao usuario a distinción entre edificios. Non pode ter valor nulo.

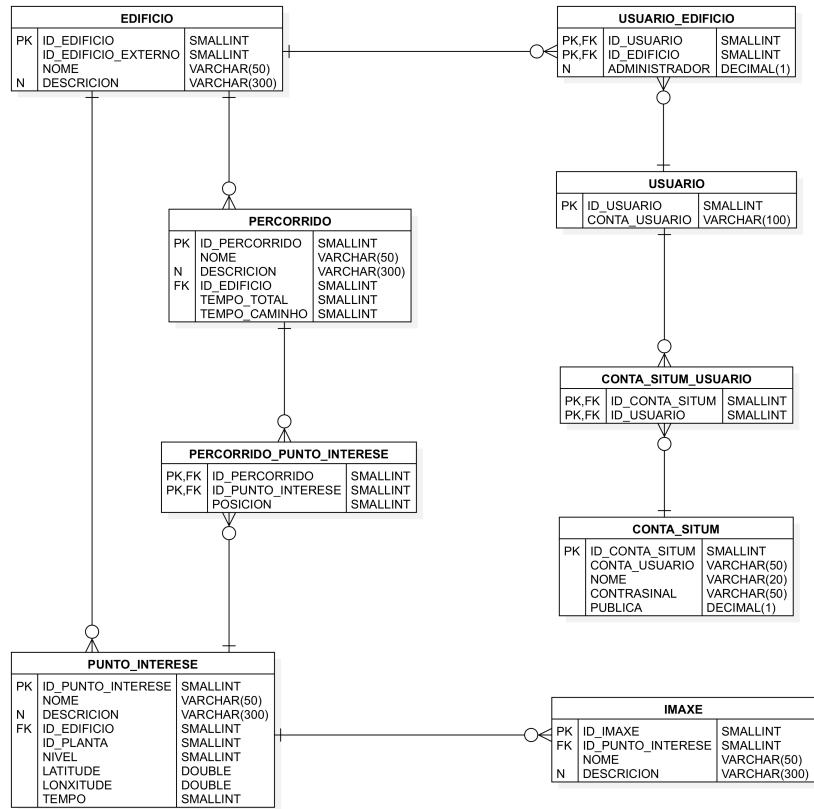


Figura 7.2: Modelo de datos da plataforma Caronte.

- DESCRICION: De tipo VARCHAR (string) con límite de 300 caracteres. Un breve texto explicativo sobre o edificio. Pode ter valor nulo.

7.2.2. USUARIO

Nesta entidade gardarase a conta de usuario de Google coa que alguéun se autentica na aplicación. Só se gardará o nome desa conta.

As columnas das que se compón este entidade son:

- ID_USUARIO: Chave primaria autoxerada da táboa. De tipo SMALLINT (short). É o identificador do usuario dentro do sistema de Caronte. Non pode ter valor nulo.
- CONTA_USUARIO: De tipo VARCHAR (string) con límite de 100 caracteres.

Correspón dese coa conta de correo de Google do usuario que se autentica na aplicación. Non pode ter valor nulo.

7.2.3. USUARIO_EDIFICIO

Nesta entidade gardarase información sobre as accións que pode levar a cabo un usuario sobre un edificio concreto. A información desta entidade é a que indica se un usuario é un xestor de contido sobre un edificio e será inserida polo Administrador do sistema.

As columnas das que se compón este entidade son:

- ID_USUARIO: Forma parte da chave primaria da entidade. Chave foránea da táboa USUARIO. De tipo SMALLINT (short). Non pode ter valor nulo.
- ID_EDIFICIO: Forma parte da chave primaria da entidade. Chave foránea da táboa EDIFICIO. De tipo SMALLINT (short). Non pode ter valor nulo.
- ADMINISTRADOR: De tipo DECIMAL con límite 1 (boolean). Indica se o usuario é xestor de contido sobre o edificio.

7.2.4. PUNTO_INTERESE

Nesta entidade gardarase información sobre os puntos de interese creados sobre un edificio e a súa localización exacta no planeta en base a coordenadas. Un punto de interese só pode pertencer a un único edificio e estará localizado nunha das súas plantas. Serán creados a través da aplicación polos Xestores de contido.

As columnas das que se compón este entidade son:

- ID_PUNTO_INTERESE: Chave primaria autoxerada da táboa. De tipo SMALLINT (short). É o identificador do punto de interese. Non pode ter valor nulo.
- NOME: De tipo VARCHAR (string) con límite de 50 caracteres. É o texto que se visualizará na aplicación e que permite ao usuario a distinción entre puntos de interese. Non pode ter valor nulo.

- **DESCRICION:** De tipo VARCHAR (string) con límite de 300 caracteres. Un breve texto explicativo sobre o punto de interese. Pode ter valor nulo.
- **ID_EDIFICIO:** Chave foránea da táboa EDIFICIO. De tipo SMALLINT (short). Permite identificar o edificio no que se atopa o punto. Non pode ter valor nulo.
- **ID_PLANTA:** De tipo SMALLINT (short). É o identificador da planta dentro do sistema de Situm. Este dato permite maior rapidez á hora de identificar os puntos dunha planta. Non pode ter valor nulo.
- **NIVEL:** De tipo SMALLINT (short). Indica o número de planta no que se atopa un punto. Non pode ter valor nulo.
- **LATITUDE:** De tipo DOUBLE. Indica a latitude do punto, o que permite unha localización exacta nun mapa. Non pode ter valor nulo.
- **LONXITUDE:** De tipo DOUBLE. Indica a lonxitude do punto, o que permite unha localización exacta nun mapa. Non pode ter valor nulo.
- **TEMPO:** De tipo SMALLINT (short). Indica o tempo en minutos de media que lle leva a un visitante admirar unha obra. Non pode ter valor nulo e o seu valor por defecto é 0.

7.2.5. PERCORRIDO

Nesta entidade gardarase información sobre os percorridos dispoñíbeis nos edificios. Un percorrido só pode pertencer a un único edificio e non está limitado a unha única planta. Está composto por varios puntos de interese que se almacenan na táboa do seguinte punto. Serán creados a través da aplicación polos Xestores de contido.

As columnas das que se compón este entidade son:

- **ID_PERCORRIDO:** Chave primaria autoxerada da táboa. De tipo SMALLINT (short). É o identificador do percorrido. Non pode ter valor nulo.
- **NOME:** De tipo VARCHAR (string) con límite de 50 caracteres. É o texto que se visualizará na aplicación e que permite ao usuario a distinción entre percorridos. Non pode ter valor nulo.

- DESCRICION: De tipo VARCHAR (string) con límite de 300 caracteres. Un breve texto explicativo sobre o percorrido. Pode ter valor nulo.
- ID_EDIFICIO: Chave foránea da táboa EDIFICIO. De tipo SMALLINT (short). Permite identificar o edificio no que se atopa o percorrido. Non pode ter valor nulo.
- TEMPO_TOTAL: De tipo SMALLINT (short). Indica o tempo en minutos de media que lle leva a un visitante realizar todo o percorrido contando co tempo que pase diante das obras. Non pode ter valor nulo e o seu valor por defecto é 0.
- TEMPO_CAMINHO: De tipo SMALLINT (short). Indica o tempo en minutos de media que lle leva a un visitante realizar todo o percorrido sen contar o tempo que pase diante das obras. Non pode ter valor nulo e o seu valor por defecto é 0.

7.2.6. PERCORRIDO_PUNTO_INTERESE

Nesta entidade gardarase a relación entre os percorridos e os puntos de interese que os componen. É obligatorio indicar a posición de cada un deses puntos dentro do percorrido xa que ten unha orde establecida. Un punto de interese só pode aparecer unha única vez nun percorrido, non se permite a súa repetición. Non hai limitación no numero de percorridos aos que pode pertencer un punto de interese.

As columnas das que se compón este entidade son:

- ID_PERCORRIDO: Forma parte da chave primaria da entidade. Chave foránea da táboa PERCORRIDO. De tipo SMALLINT (short). Non pode ter valor nulo.
- ID_PUNTO_INTERESE: Forma parte da chave primaria da entidade. Chave foránea da táboa PUNTO_INTERESE. De tipo SMALLINT (short). Non pode ter valor nulo.
- POSICION: De tipo SMALLINT (short). Indica a posición ordenada dun punto de interese dentro dun percorrido. Non pode ter valor nulo.

7.2.7. CONTA_SITUM

Nesta entidade almacénanse as distintas contas de usuario dispoñíbeis na aplicación para o acceso á plataforma de Situm. Con elas permítese a visión de distintos edificios e o seu tratamento na aplicación. As contas poden ser públicas ou privadas, precisando certo permiso o usuario para poder visualizar estas últimas. A indicación de se unha conta é publica ou privada decídeo o Administrador do sistema.

As columnas das que se compón este entidade son:

- ID_CONTA_SITUM: Chave primaria autoxerada da táboa. De tipo SMALLINT (short). É o identificador da conta de Situm. Non pode ter valor nulo.
- CONTA_USUARIO: De tipo VARCHAR (string) con límite de 50 caracteres. É o correo electrónico que utiliza Situm para autenticarse. Non pode ter valor nulo.
- NOME: De tipo VARCHAR (string) con límite de 20 caracteres. É o texto que se visualizará na aplicación e que permite ao usuario a distinción entre contas de Situm. Non pode ter valor nulo.
- CONTRASINAL: De tipo VARCHAR (string) con límite de 50 caracteres. É o contrasinal que permite a conexión ao sistema de Situm. Non pode ter valor nulo.
- PUBLICA: De tipo DECIMAL con límite 1 (boolean). Indica se a conta de Situm pode ser por calquera usuario ou en cambio debe ter permisos sobre ela. Non pode ter valor nulo.

7.2.8. CONTA_SITUM_USUARIO

Nesta entidade almacénanse as relacións entre os usuarios e as distintas contas de acceso a Situm sobre as que teñen permiso. Non almacena máis que esa relación. Se o usuario ten algunha conta asociada poderá utilizala para acceder ao sistema. Estas relacións estableceas o Administrador do sistema.

As columnas das que se compón este entidade son:

- ID_CONTA_SITUM: Forma parte da chave primaria da entidade. Chave foránea da táboa CONTA_SITUM. De tipo SMALLINT (short). Non pode ter valor nulo.

- ID_USUARIO: Forma parte da chave primaria da entidade. Chave foránea da táboa USUARIO. De tipo SMALLINT (short). Non pode ter valor nulo.

7.2.9. IMAXE

Nesta entidade almacénase a información sobre todas as imaxes subidas. Cada rexistro debe facer referencia a un punto de interese concreto. A propia imaxe non é almacenada nesta táboa, senón que se gardará no servidor. A información almacenada nesta táboa permitirá a identificación da ruta creada para recuperar a imaxe. As imaxes son engadidas polos Xestores de contido.

As columnas das que se compón este entidade son:

- ID_IMAXE: Chave primaria autoxerada da táboa. De tipo SMALLINT (short). É o identificador da imaxe. Non pode ter valor nulo.
- ID_PUNTO_INTERESE: Chave foránea da táboa PUNTO_INTERESE. De tipo SMALLINT (short). Permite identificar o punto de interese ao que fai referencia a imaxe. Non pode ter valor nulo.
- NOME: De tipo VARCHAR (string) con límite de 50 caracteres. É o texto que se visualizará na aplicación e que permite ao usuario a distinción entre imaxes. Non pode ter valor nulo.
- DESCRICION: De tipo VARCHAR (string) con límite de 300 caracteres. Un breve texto explicativo sobre a imaxe. Pode ter valor nulo.

7.3. Servidor

Nesta sección describiremos as decisións más importantes tomadas no deseño do servidor. No primeiro punto explícanse as capas nas que se dividiu o servidor e o obxectivo de cada unha delas. Nos seguintes puntos expóñense as opcións levadas a cabo para a definición do servizo web ou a transaccionalidade, entre outras.

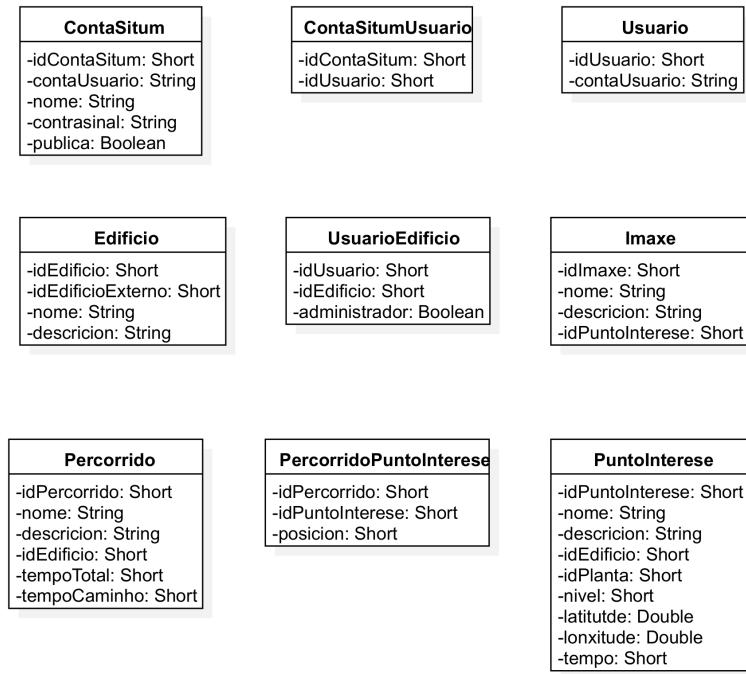


Figura 7.3: Diagrama de clases coas entidades.

7.3.1. División en capas

O servidor divídese nunha serie de capas que se diferencian no obxectivo que persegúen. Na figura 7.1 pódense ver dentro do servidor e como interactúan entre elas. As capas do servidor son as seguintes:

- **Entidade:** A representación da información recuperada da base de datos faise mediante certas clases denominadas entidades. Non teñen lóxica, só se utilizan para o almacenamento de datos. Poden utilizarse en todas as capas do servidor. Pódense observar na figura 7.3.
- **DAO:** Capa de acceso a datos. Nesta capa prodúcese a recuperación da información de base de datos, así como a súa inserción, borrado e modificación a través das entidades. Esta capa está ausente de toda lóxica de tratamiento destes datos, limítase á realización das accións comentadas, deixando os trámites ás capas superiores. Pódense observar na figura 7.4.

- Manager: Encárgase da xestión dos datos. Se no tratamento da información se precisa unha combinación entre os datos recuperados de distintos puntos é aquí onde se realiza. Nesta capa é onde se implementa a transaccionalidade, polo que se ocorre un erro durante a execución dun método dentro do manager que realiza cambios en base de datos, desfai as modificacíons previas realizadas en BD para que a información quede consistente.
- Controlador: Encárgase da comunicación a través dos servizos web. Implementa a API REST que permite o acceso aos servizos.

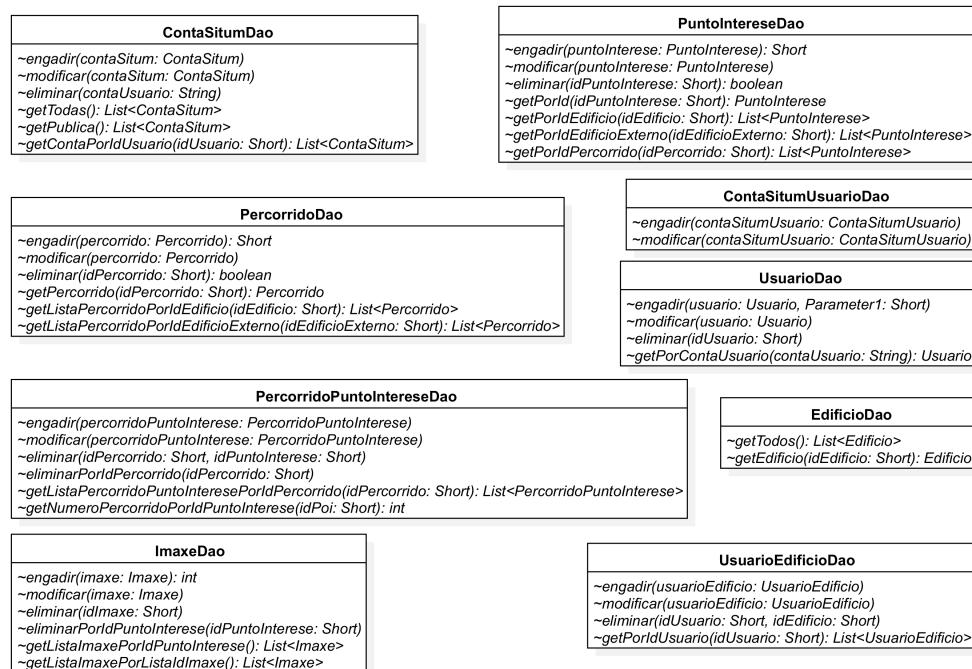


Figura 7.4: Diagrama de clases cos DAOs.

7.3.2. Transaccionalidade

Ningunha aplicación está libre de que se produza algún erro na súa execución, ben sexa por culpa de errores na implementación ou por elementos externos (caídas de conexións, errores na base de datos, etc...), polo que é preciso ter en conta estas situacíons

á hora de realizar modificacións sobre a base de datos para evitar a existencia de información errónea. Cando se realizan modificacións en serie sobre a base de datos que requieren da intervención de varias táboas pode ocorrer un erro xusto no medio, provocando que algunas táboas fosen modificadas e outras non. Neste caso romperíase a integridade dos datos. Para evitar isto, búscase un sistema que permita que se realicen esas modificacións como se fosen unha soa, e no caso de que se produza un erro, que se desfagan os cambios.

Método	Dirección	Descripción
GET	/recuperar/(idEd)/(idPoi)/(idIm)	Recupera a imaxe indicada polo id=(idIm) do poi con id=(idPoi) do edificio=(idEd)
POST	/subir	Envía unha imaxe para un POI dun edificio concreto, engadindo nome e descripción
POST	/actualizar/(idIm)/(nombre)/(desc)	Actualiza a información, nome e descripción, da imaxe indicada polo id=(idIm)
DELETE	/eliminar/(idIm)	Elimina a imaxe indicada polo id=(idIm)

Táboa 7.1: Descripcións da API Rest para o tratamiento de imaxes.

7.3.3. API REST

A API utilizada para os servizos web é REST, polo que o formato dos datos enviados e recibidos son, na súa maioría JSON. A única excepción prodúcese no envío de imaxes. As respostas producidas polos servizos tamén serán devoltas no mesmo formato. A API foi dividida en dúas seccións, por unha parte a relacionada coas imaxes (envío, recepción e modificación de datos) e por outra a relacionada co resto de elementos do sistemas (POIs, edificios, percorridos). O tratamiento de imaxes é un servizo que é susceptíbel de ser modificado para levalo a cabo por outro métodos xa que pode ter un alto consumo de datos. Desta maneira limitase o impacto que tería unha modificación neste aspecto dentro dos nosos servizos se se queren utilizar cachés propias para imaxes.

Todas as URLs comezan por */sw* mais a continuación dependen da sección á cal pertenzan: se son da información xeral levarán a continuación */museo*, mentres que as de tratamiento de imaxes levarán */imaxe*. Un exemplo de URL de cada sección:

- `http://direccionSW:porto/sw/museo/contas/(idUsuario)`
- `http://direccionSW:porto/sw/imaxe/recuperar/(idEdificio)/(idPoi)/(idImaxe)`

Os parénteses fan referencia a variábeis dentro das chamadas, texto que non é fixo senón que varía dependendo dos datos que queira solicitar o usuario.

Nas táboas poden observarse todas as chamadas dispoñíbeis dentro dos servizos web e as súas características, tanto para as imaxes (táboa 7.1) coma para o resto de información (táboa 7.2).

Método	Dirección	Descripción
GET	/edificios	Recupera todos os edificios
GET	/pois/(idEdificioExterno)	Recupera todos os POIs do edificio con id externo=(idEdificioExterno)
GET	/contas	Recupera todas as contas de Situm públicas
GET	/contas/(idUsuario)	Recupera todas as contas de Situm ás que ten acceso o usuario con id=(idUsuario)
GET	/percorridos/(idEd)	Recupera todos os percorridos do edificio con id=(idEdificio)
GET	/percorridosidexterno/(idEdEx)	Recupera todos os percorridos do edificio con id externo=(idEdEx)
GET	/ppi/(idPercorrido)	Recupera os puntos de interese do percorrido con id=(idPercorrido)
POST	/percorrido/gardar	Garda os datos dun percorrido en BD
POST	/comprobarUsuarioGoogle	Comproba que a autenticación a través da conta de Google do usuario é correcta
POST	/poi/gardar	Garda os datos dun POI en BD
DELETE	/percorrido/eliminar/(idPercorrido)	Elimina un percorrido con id=(idPercorrido). Devolve un valor indicando o éxito
DELETE	/poi/eliminar/(idPoi)	Elimina un POI con id=(idPoi). Devolve un valor indicando o éxito
GET	/imaxe/recuperar/(listaIdImaxeCSV)	Recupera a información dunha lista de imaxes en CSV

Táboa 7.2: Descripcións da API Rest para o tratamento da información xeral.

7.3.4. Intercambio de datos

O intercambio de datos entre a aplicación Android e o servidor faise en formato JSON. Na figura 7.5 pódese ver un exemplo de solicitude de creación dun punto de interese. A devolución dos datos tamén se realiza no mesmo formato. Un exemplo disto pódese observar na figura 7.6, onde se expón a resposta a unha chamada para a recuperación da información dos puntos de interese dun edificio.

```
{
  "nombre": "Gernika",
  "descripcion": "Obra de Pablo Picasso",
  "posicion": {
    "idEdificio": 3,
    "idPlanta": 1130,
    "nivel": 0,
    "latitude": 43.33301,
    "lonxitude": -8.41099
  },
  "tempo": 10,
  "listaIdImaxe": []
}
```

Figura 7.5: Exemplo dunha solicitude ao servizo web.

En ambas figuras obsérvase a estrutura que caracteriza este formato de intercambio de datos. Outra das cuestiós que nos permite observar é que a súa estrutura non coincide coa utilizada en base de datos, xa que as necesidades da aplicación non teñen por que coincidir coas decisións tomadas á hora de definir as táboas de base de datos. É por iso que se observa a agrupación da información sobre a localización física dos POIs nunha etiqueta chamada posición, que non coincide coa maneira de gardar eses valores en BD. A maiores tamén se inclúen os identificadores das imaxes almacenadas sobre cada POI, para simplificar a recuperación das imaxes na aplicación.

Para realizar a chamada ao servidor utilízase a librería spring-android-rest-template, que é o cliente para aplicacíons Android de Spring que permite a realización de chamas REST. Na sección de implementación describiranse exemplos da utilización desta librería.

Na parte do servidor tamén se utiliza Spring mediante a librería spring-mvc, que se encarga de converter automaticamente as clases a formato JSON. Para isto débe-

```
[{"punto": {"idPuntoInterese": 6, "nome": "Mona Lisa", "descripcion": "Pintura feita polo artista italiano Leonardo da Vinci mostrando unha muller cunha expresion introspectiva, lixeiramente sorrinte. E probablemente o retrato mais famoso na historia da arte.", "posicion": {"idEdificio": 2, "idPlanta": 3506, "nivel": 1, "latitude": 43.28966522216797, "lonxitude": -8.393324851989746}, "tempo": 10, "listaIdImaxe": []}, {"punto": {"idPuntoInterese": 7, "nome": "A consagracion de Napoleon", "descripcion": "Pintura de Jacques-Louis David, pintor oficial de Napoleon Bonaparte realizada entre 1805 e 1808. O cadro ten unhas impresionantes dimensons de 629 x 979 cm.", "posicion": {"idEdificio": 2, "idPlanta": 3506, "nivel": 1, "latitude": 43.289661, "lonxitude": -8.393355}, "tempo": 15, "listaIdImaxe": [3,4,5,6,7,8,9,10,11,12]}]}
```

Figura 7.6: Exemplo dunha resposta do servizo web.

se incluir a dependencia de jackson-databind e indicar nas chamadas as etiquetas `@RequestBody` e `@ResponseBody`. Igual que para as chamadas dende Android, na sección de implementación amosarase algúun exemplo desta configuración.

7.3.5. Securización das comunicacións

Tal e como se indicou no capítulo de conceptos teóricos, os servizos REST non teñen estado, polo que non se garda información ningunha entre peticións. É por isto que en cada petición se debe enviar toda a información para que se poida procesar, o que inclúe a información de securización. Para a autenticación dos servizos utilizouse o sistema Basic, que consiste no envío dun usuario e contrasinal codificados en Base64. Na cabeceira da chamada REST aparecerá algo semellante ao texto da figura 7.7.

`Authorization: Basic QWxhZGRpbjpPcGVuU2VzYW1l`

Figura 7.7: Exemplo dunha cabeceira REST securizada.

7.4. Sistema de autenticación a través de Google

Debido á necesidade de distinguir os usuarios da aplicación entre os normais e os xestores de contido houbo que integrar un sistema de autenticación na aplicación, mais non se creou un propio. Por motivos de comodidade e seguridade escolleuse a opción de realizar a autenticación dos usuarios a través de contas de Google, desta maneira non é preciso gardar información privada dos usuarios quedando estes datos do lado de Google. Ao estar destinada a aplicación ao sistema operativo Android non se obriga ao usuario a ter que crear unha conta ex profeso xa que é obligatorio o uso dunha para poder utilizar un dispositivo móvil con Android.

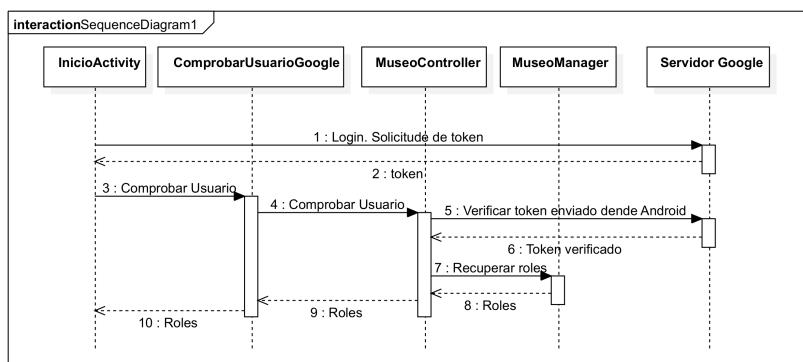


Figura 7.8: Diagrama de secuencia da autenticación a través de Google.

Para a integración do sistema de autenticación na aplicación hai que facer uso dos servizos de Google Play tanto no servidor coma na aplicación Android. É preciso configurar a aplicación para que permita a autenticación, acción que se fai a través dun formulario onde se indica o proxecto, o nome da aplicación e o seu tipo (Android, neste caso).

No diagrama da figura 7.8 pode observarse o proceso de autenticación que se explica a continuación. Unha autenticación exitosa require de varios pasos. O primeiro é a autenticación dende a aplicación Android, dende a cal se enviarán os datos da conta a Google para que os verifique. Se os datos son correctos devólvese información sobre a conta á aplicación, para posteriormente enviar un token de sesión ao servidor que permita identificar ao usuario. No seguinte paso, o servidor recolle ese token e o envía

de novo aos servizos de Google para verificar a súa integridade. Unha vez verificada, enviará a información sobre os roles dos que ese usuario dispón á aplicación Android para que poida utilizala coa conta seleccionada.

Os roles resultantes da autenticación a través da conta de Google poden permitir a modificación da información en certos edificios se ese usuario exerce como xestor de contido; noutro caso, actuará coma un usuario anónimo máis. Este rol de xestor de contido amosará un novo botón no mapa que permitirá ter acceso ao modo de edición, no que se mostrarán os botóns de creación e modificación de POIs e percorridos. Tamén dependen do usuario as contas de Situm dispoñíbeis, xa que é posíbel que algunha das contas sexan privadas en vez de públicas e se outorguen permisos individualmente.

7.5. Aplicación Android

Escolleuse unha única aplicación tanto para a consulta como para a edición para aproveitar a potencia que nos dá o mapa de interiores á hora de localizar os puntos desexados e marcalos directamente. Desta maneira pódense situar con precisión os puntos sobre o mapa mentres se percorre o edificio. Non obstante, para unha futura ampliación do sistema sería interesante permitir a configuración dende un cliente web para a creación de contido referente aos puntos e os percorridos, xa que dende un dispositivo móvil non se obtén a mesma comodidade á hora de inserir textos e proporcionar información.

Procurouse realizar un deseño moi visual mediante a utilización de moitos botóns con imaxes na actividade do mapa para un mellor aproveitamento do espazo e un estilo más clásico no resto de actividades que resultase menos rechamante pero máis cómodo. As cores principais da aplicación son o azul escuro sobre fondos brancos e maxenta coma cor secundaria.

A estrutura de paquetes da aplicación é moi simple, separando as clases en cinco paquetes principais con base *gal.caronte*:

- **activity:** Neste paquete inclúense todas as actividades utilizadas na aplicación e que se explican no seguinte punto.
- **custom:** Obxectos propios da aplicación que se utilizan para transmitir datos

entre actividades ou almacenar información recuperada do servidor. Estas últimas clases están almacenadas no paquete interno *sw*.

- **servizo:** As clases nas que se producen as conexións contra o servidor ou contra Situm atópanse neste paquete.
- **util:** Clases de utilidades para tratar cadeas de texto, solicitar permisos ou outro tipo de accións que poden ser comúns en diversas clases.
- **view:** Elementos da vista que poderían utilizarse en varias actividades, coma o selector de POIs e percorridos.

7.5.1. Actividades

Unha das primeiras decisións á hora de crear unha aplicación Android débese tomar sobre a utilización ou non de fragmentos. Apostouse por unha aplicación que utilizase varias actividades cunha única excepción, o fragmento onde se mostra o mapa de Google Maps. Esta decisión adoptouse debido á maior complexidade que adquire o código co tratamento de fragmentos e o pouco proveito que se lles sacaría nesta aplicación.

A aplicación non ten un gran número de actividades debido a que case toda a lóxica recae sobre a actividade que contén o mapa e que supón o centro de toda actividade realizada na aplicación. A continuación describimos cada unha das actividades creadas.

xiso varias cap-
as e realizar
a navegación
re elas

A actividade inicial permite a autenticación a través de Google e a selección da conta de Situm utilizada que permitirá o acceso a certos edificios. As contas dispoñíbeis estarán restrinxidas en base á conta coa que se autentique o usuario, polo que non sempre se verán as mesmas. Hai algunas que son públicas polo que estarán dispoñíbeis para todos os usuarios da aplicación. Se o usuario se autentica a través de Google pode dispoñer de permisos de xestor de contidos dentro dalgún edificio, polo que disporá de máis opcións no resto de actividades.

<https://developers.google.com/maps/documentation/android-sdk/signup>

A actividade na que se sitúa o mapa, como xa comentamos, é a principal, e ten por nome *MapaActivity*. O fragmento de Google Maps que amosa o noso mapa modificado

```
<!-- GMaps api key -->
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="@string/google_maps_key" />
```

Figura 7.9: Chave para o acceso a Google Maps.

atópase nesta actividade. Para poder utilizar Google Maps na aplicación houbo que activar unha chave na consola de Google Cloud e incluila no *AndroidManifest*. Esta chave pódese observar na figura 7.9. É a pantalla na que o usuario pasará máis tempo e onde se require amosar maior información, polo que tamén é a más complexa. A maioria das accións levadas a cabo sobre a aplicación se realizarán sobre esta actividade, como a de visualizar puntos de interese e percorridos. Se o usuario dispón de permisos de xestor de contidos dentro dun edificio, habilitarase o botón de edición que lle permitirá crear, modificar e eliminar puntos de interese e percorridos a través de novos botóns na barra principal da aplicación.

Creáronse unha serie de actividades dirixidas a amosar certa información dos puntos de interese e percorridos dos edificios cunha estrutura semellante: DetallePoiActivity e DetallePercorridoActivity. Nelas obsérvanse os datos propios deses elementos, permitindo a súa modificación e borrado nos xa existentes, e tamén a súa creación no caso de non existir previamente se o usuario ten permisos para realizar esas accións (xestor de contido).

Dende a actividade DetallePoiActivity pódense visualizar imaxes dentro dos POIs se dispoñen delas, así coma subir novas imaxes se o usuario é xestor de contido. Pódense sacar novas fotos a través da cámara ou subir imaxes dende a galería do dispositivo.

Para engadir lóxica ao mapa optouse por externalizar certos componentes coma o selector de piso e os spinners de selección para os puntos de interese e os percorridos para desta maneira non sobrecargar a actividade.

Nalgunhas das actividades utilizáronse menús contextuais para ofrecer certas opcións ao usuario. A adición ou eliminación dos puntos de interese dentro dun percorrido, a escolla entre cámara e galería á hora de subir imaxes ou a selección da imaxe a visualizar son casos nos que se optou por esta opción posto que esas accións eran demasiado

complexas para poder realizaras nun único paso.

7.5.2. Comunicación co servidor

Para acceder á información dispoñíbel no servidor utilízanse servizos web propios que devolven os datos en formato JSON. As clases que se utilizan para comunicar esta información non teñen equivalencia exacta cos VOs, polo que hai que facer conversións antes de enviala e despois de recibila. Estas clases tamén se utilizan na aplicación Android.

A comunicación co servidor realízase mediante a librería de Spring adaptada a Android: `spring-android-rest-template`. Cada chamada realízase dende unha clase distinta para identificar mellor que fai cada unha delas. No apartado de implementación amosarase a creación destas chamadas.

Tal e como se comentou no apartado de "Securización das comunicacións", as chamadas precisan do envío dun token que permita o acceso aos servizos. Na sección de implementación poderá observarse como se engaden estes datos á cabeceira da solicitude.

7.5.3. Comunicación con Situm

Todo acceso realizado sobre o sistema de Situm prodúcese dende a aplicación móvil. O noso servidor non pode ter acceso ao sistema de Situm, polo que toda a información proporcionada sobre a localización que se precise no servidor debe ser enviada pola aplicación Android. Esta comunicación que se produce co sistema de Situm é en base ás credenciais que escolla o usuario, xa sexan públicas ou privadas.

No paquete `gal.caronte.servizo.situm` atópanse as clases que realizan as chamadas aos servidores de Situm. Cada chamada permitida a Situm realízase dende unha clase distinta, igual que se fixo coas comunicacións co servidor propio, para desta maneira identificar mellor que se fai dentro de cada unha delas.

Utilízase a librería proporcionada por Situm para realizar as chamadas. O acceso a esta librería indícase no ficheiro `build.gradle` onde se fai referencia á dependencia `es.situm:situm-sdk:2.9.0@aar`. No apartado de implementación explicarase a realización

das solicitudes a Situm.

7.5.4. Permisos necesarios

A inmensa maioría de aplicacóns en Android precisan dunha lista concreta de permisos que son necesarios para unha correcta execución ou para obter unha experiencia completa. Actualmente non se require a aceptación dos permisos na instalación, senón que poden concederse individualmente durante a execución da mesma.

Débense distinguir entre os permisos para os que fan falla unha aprobación explícita dos que van implícitos na execución. O caso do permiso sobre internet é un do segundo tipo, xa que ao estar tan estendido o seu uso, nas novas versións de Android non é preciso a súa aprobación.

A continuación describiranse os motivos polos que se solicitan cada un dos permisos da aplicación. O permiso *ACCESS_FINE_LOCATION* requírese para un posicionamento axustado na utilización do mapa, polo que se non se acepta non se podería utilizar correctamente a aplicación. Os outros tres permisos solicítanse se se quere fazer uso das imaxes dentro da aplicación, xa sexa para sacar fotos de POIs (*CAMERA*), para acceder ás imaxes da galería do dispositivo (*READ_INTERNAL_STORAGE*) ou para visualizar algunha imaxe dun POI (*WRITE_EXTERNAL_STORAGE*, para gardala temporalmente).

Capítulo 8

Implementación

Neste capítulo se comentarán os aspectos más relevantes da implementación da nosa plataforma, ben sexa polo uso de librerías externas ou porque requiriron un desenvolvemento especial. Farase mención a detalles do servidor, da aplicación Android así como da autenticación a través de Google.

8.1. Servidor

Nesta sección vanse tratar todos os detalles de implementación propios do servidor, dende o acceso á base de datos ata a construcción dos servizos web. En todo o servidor utilizouse a ferramenta de Spring para configurar a transaccionalidade, a inxección de dependencias ou a creación dos servizos web.

A autenticación con Google terá a súa propia sección ao final do capítulo.

8.1.1. Acceso á base de datos

Para o acceso á base de datos utilizouse directamente JDBC sen fazer uso de ningunha ferramenta de mapeo obxecto-relacional como pode ser Hibernate. Escolleuse esta opción por ser un sistema sen moita complexidade á hora de gardar ou eliminar rexistros en base de datos e cun número de táboas non moi alto. En sistemas pequenos non se aproveitan os beneficios que poden ter estas ferramentas e desta maneira afórrase a súa configuración.

```

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
  destroy-method="close">
  <property name="driverClassName" value="org.postgresql.Driver" />
  <property name="url" value="jdbc:postgresql://direccionBD:5432/museo" />
  <property name="username" value="" />
  <property name="password" value="" />
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate">
  <constructor-arg ref="dataSource" />
</bean>

<context:property-placeholder location="classpath:queries.properties" />

```

Figura 8.1: Configuración da base de datos no ficheiro XML.

Na figura 8.1 pódese observar a configuración do data source que permite a conexión coa BD no servidor. Nel indícase o nome do driver utilizado para a conexión, así como a URL onde se atopa a base de datos. Tamén é preciso engadir o nome de usuario e o contrasinal que permiten o acceso.

Na mesma figura tamén se observa a definición do bean NamedParameterJdbcTemplate, que permite a escritura de consultas con parámetros utilizando nomes, e que se sirve do data source para lograr a conexión coa base de datos. As consultas utilizadas pola plantilla de JDBC sitúanse no ficheiro queries.properties, onde Spring as localiza para despois inxectalas directamente nos DAOs.

```

@Repository
public class ContaSatumDaoImpl implements ContaSatumDao {

```

Figura 8.2: Clase de implementación dun DAO.

Creouse unha implementación para cada un dos DAOs amosados na sección de deseño. Dentro deles fanse todas as accións permitidas sobre as táboas que representa cada VO. A implementación de cada un dos DAOs segue a mesma estrutura. Para indicarlle a Spring que estas son as clases que se utilizan para a recuperación de información da base de datos etiquétanse con `@Repository`, e dese xeito poder inxectalas na capa de xestión da información (manager). Esta etiqueta pódese ver na figura 8.2.

Na figura 8.3 pódese observar como se inxecta a plantilla de JDBC que permite a conexión coa base de datos e unha consulta, neste caso, de inserción de datos. A plantilla foi definida nun ficheiro XML como se observou na figura 8.1, e a consulta

```

    @Autowired
    private NamedParameterJdbcTemplate jdbcTemplate;

    @Value("${contaSitumDao.insertQuery}")
    private String insertQuery;

```

Figura 8.3: Inxección dunha consulta nun DAO.

para inserir datos recupérase do ficheiro queries.properties, onde se atopa marcada coa etiqueta que se inclúe dentro de `@Value`.

```

@Override
public List<ContaSitum> getContaPorIdUsuario(Short idUsuario) {
    SqlParameterSource parameters = new MapSqlParameterSource().addValue(ContaSitumUsuario.ID_USUARIO, idUsuario);
    return this.jdbcTemplate.query(this.selectPorIdUsuarioQuery, parameters, ROW_MAPPER);
}

```

Figura 8.4: Método de consulta dun DAO.

As consultas a base de datos lánzanse a través de `jdbcTemplate`, como se pode observar na figura 8.4. Neste exemplo lánzase a consulta inxectada no atributo `selectPorIdUsuarioQuery` cos parámetros que se engaden previamente, que é o identificador do usuario. Os datos gárdanse no VO grazas ao mapa definido en `ROW_MAPPER`, que se pode ver na figura 8.5.

```

private final static RowMapper<ContaSitum> ROW_MAPPER = new RowMapper<ContaSitum>() {

    @Override
    public ContaSitum mapRow(ResultSet rs, int rowNum) throws SQLException {
        Short idContaUsuario = rs.getShort(ContaSitum.ID_CONTA_SITUM);
        String contaUsuario = rs.getString(ContaSitum.CONTA_USUARIO);
        String nome = rs.getString(ContaSitum.NOME);
        String contrasinal = rs.getString(ContaSitum.CONTRASINAL);
        boolean publica = rs.getBoolean(ContaSitum.PUBLICA);
        return new ContaSitum(idContaUsuario, contaUsuario, nome, contrasinal, publica);
    }
};

```

Figura 8.5: Construción dun VO dentro do DAO.

En canto á consulta mencionada no parágrafo anterior, podemos ver como está no ficheiro de consultas na figura 8.6. Pódese apreciar a maneira de marcar o lugar onde se situarían os parámetros introducidos no método do DAO, :ID_USUARIO, neste caso.

Cando se executa a consulta substitúese esa cadea de texto polo valor inserido no método como se viu na figura 8.4.

```
contaSitumDao.selectPorIdUsuarioQuery = \
    SELECT CS.ID_CONTA_SITUM, CS.CONTA_USUARIO, CS.NOME, CS.CONTRASINAL, CS.PUBLICA \
    FROM MUSEO.CONTA_SITUM CS \
    LEFT JOIN MUSEO.CONTA_SITUM_USUARIO CSU ON (CSU.ID_CONTA_SITUM = CS.ID_CONTA_SITUM) \
    WHERE CSU.ID_USUARIO = :ID_USUARIO OR CS.PUBLICA = 1
```

Figura 8.6: Exemplo dunha consulta en SQL.

Por último, amosar como se levaría a cabo unha sentencia de inserción ou modificación de datos dentro dos DAOs. Na figura 8.7 pódese ver como se establecen os valores para a inserción dunha nova conta de Situm dentro da base de datos e como se utiliza o método *update* pasándolle a sentencia e o mapa cos parámetros.

```
@Override
public void engadir(ContaSitum contaSitum) {
    SqlParameterSource parameters = new MapSqlParameterSource().addValue(ContaSitum.CONTA_USUARIO, contaSitum.getContaUsuario())
        .addValue(ContaSitum.CONTRASINAL, contaSitum.getContrasinal())
        .addValue(ContaSitum.NOME, contaSitum.getName())
        .addValue(ContaSitum.PUBLICA, contaSitum.getPublica());
    this.jdbcTemplate.update(this.insertQuery, parameters);
}
```

Figura 8.7: Método de inserción dun DAO.

8.1.2. Transaccionalidade

A xestión transaccionalidade impleméntase na capa Manager utilizando o framework Spring, grazas á súa librería spring-tx. A súa configuración e utilización é moi sinxela, tal e como se pode observar nos exemplos. No primeiro amósase a configuración que require nos ficheiros XML de Spring. Para indicar a transaccionalidade, utilizaremos etiquetas que permitan identificar o tipo de transacción que queremos que se aplique en cada método público do manager. Non todos os métodos provocan escrituras en base de datos, polo que non será necesario indicar o tipo de transaccionalidade que provoca desfacer cambios en todos eles. O resto marcaranse como de só lectura para non sobrecargar innecesariamente o sistema.

Na figura 8.8 pódese observar a configuración da transaccionalidade nos ficheiros XML.

```

<tx:annotation-driven transaction-manager="txManager" />

<bean id="txManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource" />
</bean>

```

Figura 8.8: Configuración da transaccionalidade no ficheiro XML.

Na figura 8.9 pódese observar a configuración da transaccionalidade sobre un método no que se modifican datos.

```

@Transactional
public Short gardarPercorrido(Percorrido percorrido, List<PuntoInterese> listaPuntoInterese)

```

Figura 8.9: Configuración da transaccionalidade dun método onde se modifican datos.

Na figura 8.10 pódese observar a configuración da transaccionalidade sobre un método de só lectura.

```

@Transactional(readOnly = true)
public Edificio getEdificio(Short idEdificio)

```

Figura 8.10: Configuración da transaccionalidade dun método de só lectura.

8.1.3. Construcción dos servizos web

Para a construcción dos servizos web creáronse dous controladores distintos, tal e como se comentou na sección de deseño: un para as imaxes e o outro para o resto de información. Esta capa comunícase coa capa dos manager, detallada na subsección anterior. Para a súa implementación usouse unha librería específica de Spring para a construcción de servizos web: Spring MVC.

Na figura 8.11 pódese observar como se configura un controlador do servizo. Coa etiqueta `@RestController` identifícanse as clases que actúan como controladores de servizos web REST, ao mesmo tempo que indica que o valor de retorno dos métodos vai no corpo da response. Dentro destas clases inclúense os métodos que estarán publicados no servizo coas súas propias anotacións. A continuación veranse distintos exemplos dos

```

@RestController
@EnableWebMvc
public class MuseoControllerImpl implements MuseoController {

```

Figura 8.11: Exemplo de etiquetas sobre o servizo.

tipos de métodos que se publican no noso servizo.

```

@Override
@GetMapping("/percorridos/{idEdificio}")
public List<PercorridoCustom> getPercorridoEdificio(@PathVariable short idEdificio) {
    List<Percorrido> lista = this.museoManager.getListaPercorridoPorIdEdificio(idEdificio);
    return convertirPercorridoPercorridoCustom(lista);
}

```

Figura 8.12: Exemplo de método Get con parámetro do servizo.

O primeiro método a observar é de tipo GET, é dicir, de recuperación de información. No caso da figura 8.12 tense unha etiqueta `@GetMapping` coa que se indica o seu tipo e a continuación a URL que activa ese método. Nas chamadas REST pódense introducir variábeis na propia URL, como ocorre neste caso co parámetro `idEdificio`. Para poder utilizar esta variábel débese indicar como parámetro dentro do método e anotala coa etiqueta `@PathVariable` e nomeala do mesmo xeito.

```

@Override
@RequestMapping(value = "/percorrido/gardar", method = RequestMethod.POST, produces = MediaType.APPLICATION_JSON_VALUE)
public Short gardarPercorrido(@RequestBody GardarPercorridoParam gardarPercorridoCustom) {
    PercorridoCustom percорridoCustom = gardarPercorridoCustom.getPercorrido();
    Percorrido percорrido = new Percorrido(percорridoCustom.getIdPercorrido(), percорridoCustom.getName(),
        percорridoCustom.getDescription(), percорridoCustom.getIdEdificio(), percорridoCustom.getTotalTime(),
        percорridoCustom.getTravelTime());
    List<PuntoInterese> listaPoi = convertirPuntoIntereseCustomPuntoInterese(gardarPercorridoCustom.getListPoi());
    return this.museoManager.gardarPercorrido(percорrido, listaPoi);
}

```

Figura 8.13: Exemplo de método Post do servizo.

Outro tipo de métodos dentro do noso servizo web é o POST, utilizado para enviar información ao servidor e realizar modificacóns. Na figura 8.13 obsérvase un exemplo deste tipo. Pasar parámetros a través da URL non é a única maneira de enviar datos a un servizo REST, xa que se pode utilizar o corpo da request coma neste caso. Spring MVC interpreta o corpo da request coma un JSON e tradúceo a un obxecto do servidor automaticamente xa que está construído cos mesmos atributos. Para indicar que o

parámetro provén do corpo da request utilízase a etiqueta `@RequestBody`.

```
@Override
@DeleteMapping("/poi/eliminar/{idPoi}")
public Boolean eliminarPuntoInterese(@PathVariable Short idPoi) {
    return this.museoManager.eliminarPuntoInterese(idPoi);
}
```

Figura 8.14: Exemplo de método Delete do servizo.

Por último, quedan os métodos DELETE, utilizados para eliminar información do servidor. Na figura 8.14 pode observarse un método deste tipo utilizado no noso servizo. Nel pode verse como recibe un parámetro pola URL e como devolve un boolean no corpo da response indicando se houbo éxito no borrado do punto de interese.

8.1.4. Autorización nos servizos

Tal e como se indicou no apartado de deseño de "Securización das comunicacións", implementouse un sistema de autenticación básica sobre os servizos web, de tal maneira que non se permite a chamada a eses servizos se non se inclúe na cabeceira un token para a autenticación. Este token constrúese mediante a codificación en Base64 dun nome de usuario e contrasinal que son comúns ás chamadas. A configuración desta securización realiza no ficheiro web.xml, onde se inclúe o código da figura 8.15.

Como se pode observar na figura, securízanse as chamadas GET, DELETE, POST e PUT, que son as básicas nos servizos REST. Calquera chamada que non teña o token de autenticación correcto será rexitada sen entrar no servizo.

8.2. Aplicación Android

Nesta sección trataranse os aspectos más importantes da implementación na aplicación Android.

8.2.1. Solicitud de permisos

A aplicación Android require de certos permisos para o seu correcto funcionamento. Actualmente non é preciso aceptar o uso destes permisos na propia instalación

```

<!-- Directivas de seguridade -->
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Privado</web-resource-name>
        <url-pattern>/*</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
        <http-method>PUT</http-method>
        <http-method>DELETE</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>aplicacion</role-name>
    </auth-constraint>
</security-constraint>

<!-- Autenticacion basica -->
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>CaronteRealm</realm-name>
</login-config>

<security-role>
    <role-name>aplicacion</role-name>
</security-role>

```

Figura 8.15: Configuración da autenticación básica no servidor.

do APK, senón que se permite a solicitude individual dentro da aplicación, cando se precisan por primeira vez. Os permisos solicitados pola aplicación débense indicar no AndroidManifest.xml e pódense ver na figura 8.16.

```

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />

```

Figura 8.16: Permisos necesarios na aplicación Android.

Non faría falla indicar a solicitude do permiso para o acceso a internet nas versións actuais de Android, mais indícase para os dispositivos que utilicen versións antigas. Deixou de ser preciso aceptar este permiso debido a que a inmensa maioría de aplicacións fan uso de internet.

Antes da realización de calquera acción que requira permisos especiais é obligatorio comprobar se a aplicación ten acceso a ese permiso, xa que o usuario pode revocalo en calquera momento. No caso de que non o teña, aparecerá un diálogo na pantalla que

```

boolean permisoConcedido = false;
if (ContextCompat.checkSelfPermission(actividad, permiso) != PackageManager.PERMISSION_GRANTED) {
    Log.i(TAG, StringUtil.creaString( ...cadaes: "Sen permiso para ", permiso, ". Procedemos a solicitalo"));
    //Se debemos explicar a solicitud do permiso creamos un aviso e realizamola
    if (ActivityCompat.shouldShowRequestPermissionRationale(actividad, permiso)) {
        PermisosUtil.RationaleDialog.newInstance(codigoSolicitud, finalizar, permiso)
            .show(actividad.getSupportFragmentManager(), tag: "dialog");
    }
    //Se non precisamos explicacion para a solicitud, realizamola directamente
    else {
        ActivityCompat.requestPermissions(actividad, new String[]{permiso}, codigoSolicitud);
    }
}
else {
    Log.i(TAG, StringUtil.creaString( ...cadaes: "Permiso ", permiso, " xa obtido. Continuamos co proceso"));
    permisoConcedido = true;
}

```

Figura 8.17: Comprobación e solicitude de permisos necesarios na aplicación Android.

pregunte ao usuario se desexa permitir esa acción. Se acepta, continuará a execución sen problema, se non o acepta e o permiso era imprescindíbel, a aplicación deixará de funcionar. A comprobación e solicitude dos permisos realizanse mediante chamadas específicas proporcionadas polas propias librerías de Android, polo que apenas require esforzo por parte do programador. Un exemplo de código pódese ver na figura 8.17.

<https://developer.android.com/training/permissions/requesting>
419

8.2.2. Acceso a Situm

Neste punto revisarase a configuración dos servizos de Situm e por outra parte, a solicitude de información sobre os edificios.

Configuración da localización

O primeiro paso para configurar o acceso a Situm é engadir a dependencia coa súa librería tal e como se indicou na sección de deseño. Unha vez se ten esa dependencia, pódese facer uso de certas clases e métodos para poder acceder aos servizos de Situm.

Para inicializar o SDK de Situm débese realizar unha chamada específica dentro do código e posteriormente indicar o usuario e o contrasinal co cal se quere acceder aos servizos. Estas accións pódense ver na figura 8.18.

Unha vez inicializado, o seguinte paso sería a activación da localización en interiores.

```

SatumSdk.init( context: this);
SatumSdk.configuration().setUserPass(nomeConta, contrasinalConta);
this.locationManager = SitumSdk.locationManager();

```

Figura 8.18: Inicialización do SDK de Situm.

```

this.locationListener = new LocationListener() {

    @Override
    public void onLocationChanged(@NonNull final Location location) {
        //Código cando cambia a localización
    }

    @Override
    public void onStatusChanged(@NonNull LocationStatus locationStatus) {
        //Código cando cambio o estado da localización
    }

    @Override
    public void onError(@NonNull Error error) {
        //Código cando se produce un erro
    }
};

```

Figura 8.19: Creación do location manager de Situm.

Para realizar isto débese solicitar esta activación aos servidores de Situm, mais antes de realizar este paso débese comprobar se a aplicación dispón do permiso de localización, imprescindíbel para o uso da aplicación. Mediante a creación dun listener específico permítense a resposta á información devolta por parte de Situm no referente á posición. Este listener (na figura 8.20) dispón de tres métodos para reaccionar ante cambios de posición, cambios de estado ou erros no servizo. O método principal ao que máis importancia lle debemos dar é ao de cambio de posición, xa que é o que máis vai modificar o estado da nosa aplicación. Devolve a información da posición actual do usuario: edificio, piso, coordenadas, orientación, entre outras. O segundo método informa sobre o estado actual do sistema, ben se está inicializado, en proceso de arranque, se o usuario non se atopa nun edificio, etcétera. Por último tense o método que recibe os erros de Situm que provoquen a detención do posicionamento.

Cando se dispoña do listener, pódese iniciar o posicionamento mediante o código amosado na figura 8.20. A partir dese momento comenzarán a chegar notificacións no listener definido no punto anterior.

```
LocationRequest locationRequest = new LocationRequest.Builder()
    .build();
this.locationManager.requestLocationUpdates(locationRequest, locationListener);
```

Figura 8.20: Activación do posicionamento de Situm.

Acceso aos servizos de Situm

Aínda que a maioría dos datos dos que ten que facer uso a aplicación proveñen do noso propio servidor, tamén se debe recuperar certa información dos servidores de Situm. As chamadas que se deben realizar aos servidores de Situm son as seguintes:

- Recuperación da lista cos edificios aos que ten acceso esa conta (nada máis establecer a conta de Situm).
- Recuperación da lista dos niveis dun edificio concreto (cando se accede a ese edificio).
- Recuperación dos mapas dos niveis dun edificio que se amosarán enriba do fragmento de Google Maps.
- Recuperación da ruta a un punto concreto do mapa. Se as rutas están configuradas en Situm e nos atopamos fisicamente dentro dun edificio, a aplicación permite solicitar rutas para dirixirse ao punto desexado, amosando as direccións que debe seguir o usuario.

Para o acceso aos servizos de Situm optouse por illar as conexións en clases distintas para identificar mellor a lóxica levada a cabo en cada unha delas. Estas clases atópanse no paquete gal.caronte.servizo.situm. Os accesos aos servidores de Situm realizáronse mediante o uso da SDK de Situm na súa versión 2.9.0.

A conexión realizaase mediante o Communication Manager de Situm, un xestor que se obtén mediante unha chamada específica. Unha vez se ten ese xestor, pódese solicitar a información desexada aos servidores de Situm mediante os métodos que dispón. Un exemplo destas chamadas pode verse na figura 8.21, onde se recuperan os niveis dun edificio concreto.

```

SatumSdk.communicationManager().fetchFloorsFromBuilding(edificioSatum, new Handler<Collection<Floor>>() {
    @Override
    public void onSuccess(Collection<Floor> listaPiso) {
        if (hasCallback()) {
            callback.onSuccess(listaPiso);
        }
        clearCallback();
    }

    @Override
    public void onFailure(Error error) {
        Log.e(TAG, error.getMessage());
        if (hasCallback()) {
            callback.onError(error);
        }
        clearCallback();
    }
});

```

Figura 8.21: Chamada para a recuperación dos niveis dun edificio en Situm.

8.2.3. Acceso ao servidor

Ao igual que se explicou no punto anterior para as conexións con Situm, as conexións co servidor propio da aplicación tamén se realizaron en clases illadas para non mesturar o código de diversas solicitudes. As chamadas son asíncronas polo que se poden realizar varias á vez e continuar coa execución da aplicación normalmente.

As chamadas ao servidor propio son distintas na súa realización. Débese indicar a dirección e incluír os datos de autorización na cabeceira en cada unha delas. Non é a única información que se deber proporcionar ao lanzar esas solicitudes, senón que tamén se ten que indicar en que formato converter os datos (JSON) e en que tipo de dato converter a resposta.

```

public class RecuperarEdificio extends AsyncTask<Void, Void, List<EdificioCustom>> {

```

Figura 8.22: Construcción dunha clase de acceso ao servidor de Caronte.

Na figura 8.22 pódese ver o formato das clases utilizadas para a comunicación co servidor. Deben estender a clase *AsyncTask* e indicar os tipos dos datos enviados ao servidor (primeiro dos xenéricos) e os datos que se recibirán (terceiro dos xenéricos). Neste exemplo non se envían parámetros na chamada e se recibe unha lista de edificios, que se verán nas seguintes figuras.

```

@Override
protected List<EdificioCustom> doInBackground(Void... params) {

    final String url = StringUtil.creaString(this.inicioActivity.getString(R.string.direccion_servidor),
        this.inicioActivity.getString(R.string.direccion_servizo_recuperar_edificios));
    RestTemplate restTemplate = new RestTemplate();
    restTemplate.getMessageConverters().add(new MappingJackson2HttpMessageConverter());

    HttpHeaders headers = new HttpHeaders();
    headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
    headers.setAuthorization(new HttpBasicAuthentication(this.inicioActivity.getString(R.string.usuario_sw),
        this.inicioActivity.getString(R.string.contrasinal_sw)));
    HttpEntity<String> entity = new HttpEntity<>(headers);

    ResponseEntity<Object> response = restTemplate.exchange(url, HttpMethod.GET, entity, Object.class);
    Object resource = response.getBody();

    return new ObjectMapper().convertValue(resource, new TypeReference<List<EdificioCustom>>() { });
}

```

Figura 8.23: Método para a realización dunha chamada ao servidor de Caronte.

O método dende o cal se realiza a chamada pódese observar na figura 8.23. O primeiro punto a ter en conta é o tipo dos parámetros que recibe o método: Void; iso indica que este método non recibe ningún parámetro para enviar ao servidor mais si se pode ver que ten un tipo de retorno, unha lista de edificios que se tratarán despois. Pode apreciarse a construcción da URL á que se vai realizar a chamada, o convertedor da mensaxe ao formato desexado (JSON), a cabeceira onde se indica o formato e os datos para a autorización básica da chamada.

As últimas sentencias do método son a propia chamada ao servidor onde se indica a URL, o tipo de chamada (GET), as entidades da solicitude (neste caso só a cabeceira) e o tipo de dato que devolve o método. A continuación poderíanse indicar os parámetros que se queren enviar coa petición separados por comas, pero nesta chamada non hai ningún. Como esta chamada tarda tempo en devolver os datos desexados semella que o fluxo queda detido neste punto, mais como xa se indicou antes, a chamada é asíncrona e pode continuar a execución da aplicación con normalidade. O último paso sería converter os datos recibidos ao tipo de obxecto desexado.

Para poder facer uso dos datos recibidos, existe outro método nestas clases de acceso ao servidor, e é o método *onPostExecute*, que recibe como parámetro o tipo de dato devolto no método *doInBackground*. Cando remata a execución da chamada os datos

```

@Override
protected void onPostExecute(List<EdificioCustom> listaEdificio) {
    this.inicioActivity.setListaEdificio(listaEdificio);
}

```

Figura 8.24: Método para o tratamento de datos devoltos por unha chamada ao servidor de Caronte.

acabarán chegando a este método dende onde se poderán realizar as accións desexadas con eles.

8.3. Autenticación

Neste punto verase o proceso de autenticación en máis profundidade, revisando as diferentes chamadas necesarias tanto dende a aplicación Android coma dende o servidor contra os servizos de Google para verificar a identidade do usuario.

O primeiro punto a ter en conta é a configuración necesaria para permitir a autenticación dende a aplicación. Débese incluír a dependencia con '*com.google.android.gms:play-services-auth:12.0.1*' no build.gradle a nivel de aplicación e incluír o ficheiro google-services.json específico de Caronte que se pode conseguir despois de dar de alta o proxecto na consola de desenvolvemento de Google. Dende esta mesma consola pódense conseguir as credenciais que permiten a conexión cos servizos de Google que utilizaremos a continuación, na conexión dende a aplicación Android e dende o servidor.

<https://console.developers.google.com/apis/credentials>

uir referencia a

<https://developers.google.com/identity/sign-in>

ndroid/start-
grating

<https://developers.google.com/android/guides/client>

O primeiro paso para permitir o acceso a un usuario concreto na aplicación é a obtención das credenciais de autenticación do usuario. Para lograr isto, temos o código da figura 8.25. Pulsando sobre o botón de login lanzarase unha actividade propia de Google que permite a autenticación dentro da aplicación cunha conta de Google.

O seguinte paso será o tratamento da información recibida cando a actividade lanzada no punto anterior remate correctamente. Pódese ver na figura 8.26 o tratamento

```
GoogleSignInOptions gso =
    new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken("937334461735-51d5hochfmdj5hour3tdh0cbigtobtjo.apps.goog...")  

        .requestEmail()
        .build();

this.apiClient = new GoogleApiClient.Builder( context: this)
    .enableAutoManage( fragmentActivity: this, onConnectionFailedListener: this)
    .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
    .build();

this.botonLogin = findViewById(R.id.boton_login);
this.botonLogin.setOnClickListener((v) -> {
    showProgressDialog();
    Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(InicioActivity.this.apiClient);
    startActivityForResult(signInIntent, RC_SIGN_IN);
});
```

Figura 8.25: Código para a autenticación con Google na aplicación.

dos datos. O punto principal é o lanzamento dunha chamada ao servidor de Caronte pasando como parametro un token enviado por Google para verificar a conta.

Na figura 8.27 pódese ver o código do servidor co cal se verifica a conta coa que se acaba de autenticar o usuario. Utilízase a clase GoogleIdTokenVerifier para verificar o token recibido na aplicación Android. Se o token é válido, devólvese a información do usuario xunto cos seus roles á aplicación Android, se ten algúns. Se é a primeira vez que o usuario accede ao sistema, gardarase a súa conta na base de datos.

Para que funcione a autenticación desde unha aplicación descargada de Google Play débese incluír a chave do certificado co que se firma a aplicación na consola para desenvolvedores de Google.

Para publicar a aplicación na Play Store
<https://developer.android.com/studio/publish/app-signing>

```
private void handleSignInResult(GoogleSignInResult result) {
    if (result.isSuccess()) {
        Toast.makeText( context: this, "Usuario logueado", Toast.LENGTH_SHORT).show();
        //Usuario logueado
        GoogleSignInAccount acct = result.getSignInAccount();
        if (acct != null) {
            String idToken = acct.getIdToken();
            this.nomeMostrar = acct.getDisplayName();

            this.tvGoogle.setText(acct.getEmail());
            this.tvNomeGoogle.setText(this.nomeMostrar);

            ComprobarUsuarioGoogle comprobarUsuarioGoogle = new ComprobarUsuarioGoogle();
            comprobarUsuarioGoogle.setInicioActivity(this);
            comprobarUsuarioGoogle.execute(idToken);

        }
        actualizarVista( loginCorrecto: true);
    }
    else {
        Toast.makeText( context: this, "Usuario no logueado" + result.getStatus().toString(),
            Toast.LENGTH_SHORT).show();
        //Usuario no logueado --> Desconectado
        actualizarVista( loginCorrecto: false);
        hideProgressDialog();
    }
}
```

Figura 8.26: Código para la autenticación con Google en la aplicación (2).

```
log.info(StringUtil.createString("Iniciase a comprobacion do usuario de Google co token: ", idTokenString));

HttpTransport httpTransport = GoogleNetHttpTransport.newTrustedTransport();
JsonFactory jsonFactory = JacksonFactory.getDefaultInstance();

GoogleIdTokenVerifier verifier = new GoogleIdTokenVerifier.Builder(httpTransport, jsonFactory)
    //Id do cliente da aplicacion para acceder a Google
    .setAudience(Collections.singletonList("937334461735-5ld5hochfmdj5hour3tdh0cbigtobtjo.apps.googleusercontent.com"))
    .build();

GoogleIdToken idToken = verifier.verify(idTokenString);
if (idToken != null) {
    Payload payload = idToken.getPayload();
    boolean emailVerified = Boolean.valueOf(payload.getEmailVerified());
    if (emailVerified) {
        String userId = payload.getSubject();
        log.info(StringUtil.createString("User ID: ", userId));

        // Get profile information from payload
        String email = payload.getEmail();

        //Recupera a conta de usuario a traves dun email. Se non existe a conta, crea a
        Usuario usuario = this.museoManager.getUsuario(email);
        resposta.setIdUsuario(usuario.getIdUsuario());
        resposta.setContaUsuario(email);

        List<UsuarioEdificio> listaUsuarioEdificio = this.museoManager.getListaUsuarioEdificioPorIdUsuario(usuario.getIdUsuario());
        for (UsuarioEdificio ue : listaUsuarioEdificio) {
            if (ue.getAdministrador()) {
                resposta.getListaIdEdificioAdministrador().add(ue.getIdEdificio());
            }
        }
    }
    else {
        log.info("O email non foi verificado");
    }
}
else {
    log.error("Token invalido");
}
```

Figura 8.27: Código para a autenticación con Google na aplicación (e 3).

Capítulo 9

Probas

Neste capítulo detallaremos as probas realizadas para validar o noso sistema, tanto ao finalizar cada sprint coma ao rematar o desenvolvemento. Tamén se terán en conta aquelas ferramentas utilizadas para localizar e ter un maior control sobre os problemas da aplicación.

9.1. Probas de validación de cada sprint

Ao longo de todo o desenvolvemento do proxecto realizáronse probas ao finalizar cada sprint, ca fin de identificar posíbeis problemas introducidos sexa sobre as funcionalidades novas ou nas xa implementadas. A partir do terceiro sprint puidéronse realizar probas directamente sobre a aplicación pois nese momento xa se dispoña dunha implementación básica que o permitía.

Estas probas realizanse utilizando a aplicación en condicións normais, tanto nun uso correcto coma nun uso erróneo para comprobar que non se producen situacións non controladas. Este uso erróneo consiste en intentar facer accións non permitidas, deixar sen cubrir certa información á hora de gardar datos, cambiar a pantalla rapidamente ou antes de que remate unha acción, etc...

As validacións destas probas realizáronse a dous niveis: a comprobación dun funcionamento correcto da aplicación a nivel de usuario e a revisión dos datos introducidos, modificados e eliminados da base de datos.

Bloqueos en tiempo real				
Últimos 60 días	Todas las versiones de Android	1.0.12 (12)	Mostrar ocultos	
Clúster	Informes	Usuarios afectados	Último informe	
Nuevo en la versión 12 java.lang.NullPointerException en gal.caronte.activity.MapaActivity.iniciarGuiado	6	1	11 de may. 1:03	⋮
Nuevo en la versión 12 java.lang.IndexOutOfBoundsException en gal.caronte.activity.MapaActivity.iniciarGuiado	3	1	11 de may. 22:32	⋮
Nuevo en la versión 12 java.lang.NullPointerException en gal.caronte.servizo.situm.RecuperarRuta.recuperarRuta	2	1	11 de may. 2:31	⋮
Nuevo en la versión 12 java.lang.IllegalStateException en android.widget.RelativeLayout\$DependencyGraph.getSortedViews	1	1	13 de may. 18:39	⋮
Nuevo en la versión 12 java.lang.NullPointerException en gal.caronte.servizo.situm.RecuperarRuta.recuperarRuta	1	1	13 de may. 18:39	⋮
Nuevo en la versión 12 java.lang.NullPointerException en gal.caronte.servizo.situm.RecuperarRuta.recuperarRuta	1	1	13 de may. 18:39	⋮
Nuevo en la versión 12 java.lang.ArrayIndexOutOfBoundsException en gal.caronte.activity.MapaActivity.avanzarPoiGuiado	1	1	15 de may. 2:39	⋮

Página 1 de 1

Figura 9.1: Pantalla coas incidencias na consola de Google Play.

9.2. Consola de Google

Ao finalizar o terceiro sprint que xa supoña unha aplicación usábel, aínda que limitada, procedeuse á publicación da aplicación en Google Play. Durante varios sprints foise publicando en modo Probas internas, sen permitir un acceso libre á mesma. Nesta modalidade permítese a proba dun número determinado de usuarios que se indican a través das súas contas de Google. Grazas a este sistema pódense recompilar os distintos errores que se produzcan así como información sobre o dispositivo no que ocorren para poder dar unha solución con maior eficacia. Despois do sexto sprint publicouse como beta aberta para poder acceder a un número maior de usuarios de proba. A publicación xera unha ligazón á aplicación que se pode compartir para que os usuarios a poidan

descargar.

Dende a consola de Google tense acceso a información de todo tipo referente á aplicación, coma o número de descargas de cada versión publicada, o número de usuarios coa aplicación instalada, así como todos os envíos dos errores producidos nos terminais de proba (figura 9.1), polo que resulta unha ferramenta moi útil para a depuración das incidencias atopadas polos usuarios que proban.

9.3. Probas de rendemento

As probas de rendemento realizáronse pensando no servidor do noso sistema e no servizo de Situm. Nun principio non se conta con que haxa problemas con ningún dos dous elementos, posto que o primeiro está despregado en Amazon Web Services e no caso de ter máis usuarios concurrentes do habitual disporíase do ancho de banda necesario; e o segundo xa está en uso en moitos puntos do planeta.

Para levar a cabo as probas, coordinouse o acceso ao sistema por parte de diversos usuarios de proba para realizar todo tipo de accións dentro da aplicación, tanto de usuario normal coma de xestor de contido. En ningún momento se observou lentitude na aplicación e en todos os casos permitiu as accións sen amosar errores, incluso á hora de enviar e recibir imaxes o servidor.

9.4. Enquisa usuarios

Ao finalizar o desenvolvemento da sistema elaborouse unha breve enquisa para a valoración da aplicación Android entre os usuarios de proba. A idea da enquisa é que fosen preguntas breves e concisas, para que as persoas non dubidasen sobre a cuestión que se preguntaba. O sistema de puntuación é numérico, dende o 1 que indica o nivel máis baixo ata o 5 que indica o nivel máis alto.

As preguntas están divididas en catro bloques:

- Usabilidade: cuestións sobre a facilidade de uso.
- Interface: preguntas sobre o atractiva que pode resultar visualmente e se está ben

deseñada.

- Funcionalidade: cuestiós sobre a utilidade da aplicación.
- Rendemento: preguntas sobre a velocidade e robustez.

Pregunta	Valoración media
USABILIDADE	
É doada de usar?	
É intuitiva?	
Require pouco tempo de aprendizaxe?	
INTERFACE	
O deseño é atractivo?	
As accións dispoñíbeis están claras?	
As accións dispoñíbeis están accesíbeis?	
FUNCIONALIDADE	
É útil?	
Recomendaríaslla a outra persoa na mesma situación?	
As súas funcionalidades son únicas (non se atopan noutra aplicación)?	
RENDEMENTO	
É estábel (sen errores)?	
É rápida nas súas accións?	

Táboa 9.1: Resultados da enquisa a usuarios.

Na táboa 9.1 poden observarse as preguntas realizadas aos usuarios de proba e a valoración media de cada unha delas. A enquisa foi respondida por 10 persoas, sendo a maioría delas xente á marxe do mundo do desenvolvemento de software.

oración da en-

ca

Capítulo 10

Conclusións e traballo futuro

Este capítulo está adicado a resumir as principais conclusión deste proxecto así coma o traballo futuro.

10.1. Conclusións

Neste traballo proponse unha plataforma para a guía de museos que consta de dúas partes: un servidor e unha aplicación Android. Ademais, empregamos o SDK e os servidores de Situm Technologies S.L. para determinar a posición dentro dun edificio.

As principais características da nosa plataforma son:

- Modelo de datos flexible: puntos de interese (POI) e percorridos de visita (lista de POIs). Tempo estimado e outros datos de interese como poden ser imaxes.
- Autenticación de usuarios para realizar tarefas de mantemento e edición de contidos.
- Usuarios anónimos para facilitar o uso da aplicación.
- Servidor na nube: procura escalabilidade, facilita o mantemento, permite axuste automático dos recursos empregados ao uso que se fai deles, etc.
- Aplicación Android intuitiva e fácil de empregar para mostrar información do museo: mapas de cada planta, POIs, percorridos de visita, etc.

- Localización Situm para calcular a posición dentro dos museos (implica unha calibración previa para cada planta dos edificios, por parte do administrador da plataforma ou do xestor de contidos do museo).
- Autenticación vía Google para identificación dos usuarios privilexiados do noso sistema.
- Edición de datos (POI e percorridos de visita) dentro da propia aplicación Android.

10.2. Traballo futuro

Este proxecto resolve os requisitos formulados inicialmente, pero tamén abre novas vías e retos para o futuro:

- O principal punto a tratar nun futuro debería ser a creación dun cliente web que permita a administración do sistema e a inserción de datos sobre os puntos de interese e os percorridos por parte dos xestores de contido. É innegábel a potencia e a comodidade de crear puntos directamente sobre o mapa, pero a experiencia dos xestores á hora de engadir información sobre os elementos directamente dentro do dispositivo móvil non é moi amigábel. Tamén é innegábel a incomodidade dos administradores do sistema de realizar todas as configuracións a través dun sistema xestor de base de datos.
- Determinar automaticamente onde están máis tempo os visitantes no museo. Desta maneira poderíanse axustar mellor os tempos almacenados en base de datos e observar se sería preciso crear engadir algúns puntos de interese aos percorridos utilizados.
- Axustar automaticamente o tempo necesario para a realización dos percorridos e o tempo adicado a cada punto de interese.
- Identificar as rutas más empregadas e ordenalas por maior interese para que poidan ser utilizadas polos usuarios menos expertos.

- Controlar a orde na que os usuarios realizan visitas para a creación de percorridos en base a esas rutas.
- Permitir aos xestores de contido engadir máis información referente aos puntos de interese e aos percorridos, facendo máis rica a experiencia de uso para os usuarios.
- Outro punto interesante sería permitir a cada usuario que indicase certos datos persoais para adaptar os datos dos puntos de interese e dos percorridos. Amosar máis información de carácter profesional a un licenciado en belas artes, en contraposición a unha información máis simple e accesíbel para un estudiante de primaria que accede por primeira vez a un museo.

Apéndice A

Casos de uso

Neste apéndice incluiremos as táboas que describen os casos de uso da aplicación:

IDENTIFICADOR	Autenticarse
Nome	Autenticarse
Descripción	Accede á aplicación introducindo as credenciais de usuario
Actor	Usuario Android
Precondicións	Non debe estar autenticado. Non é preciso estar rexistrado xa que se accede coas credenciais de Google (necesarias para Android)
Poscondicións	Estará autenticado na aplicación e terá acceso as súas funcións
Escenario	Na pantalla inicial terá a posibilidade de pulsar un botón para autenticarse cunha conta de Google

Táboa A.1: Caso de uso Autenticarse.

IDENTIFICADOR	Logout
Nome	Logout
Descripción	Deixa de utilizar as credencias dunha conta de Google
Actor	Usuario Android
Precondicións	O usuario debe estar autenticado
Poscondicións	O usuario xa non estará autenticado
Escenario	Na pantalla inicial terá a posibilidade de pulsar un botón para facer Logout se xa estaba autenticado

Táboa A.2: Caso de uso Logout.

IDENTIFICADOR	Recuperar contas Situm
Nome	Recuperar contas Situm
Descripción	Recupera as contas utilizadas en Situm dun usuario
Actor	Usuario Android
Precondicións	Accede á pantalla inicial da aplicación. Pode estar autenticado ou non
Poscondicións	Amosará as distintas contas ás que teña acceso
Escenario	Ao abrir a pantalla de inicio e ao autenticarse

Táboa A.3: Caso de uso Recuperar contas Situm.

IDENTIFICADOR	Acceder mapa
Nome	Acceder mapa
Descripción	Amosa o mapa de Google Maps
Actor	Usuario Android
Precondicións	Estar na pantalla inicial da aplicación. Pode estar autenticado ou non
Poscondicións	Amosará o mapa de Google Maps
Escenario	Na pantalla inicial, pulsar sobre o botón de Acceder

Táboa A.4: Caso de uso Acceder mapa.

IDENTIFICADOR	Amosar mapa edificio
Nome	Amosar mapa edificio
Descripción	Amosa o mapa dun edificio superposto a Google Maps na pantalla principal
Actor	Usuario Android
Precondicións	Estar na pantalla principal e ter permiso para ver ese edificio
Poscondicións	Amosará o mapa. Permite realizar accións sobre o edificio
Escenario	Na pantalla principal, pinchar sobre a situación dun edificio no mapa ou atoparse fisicamente nel

Táboa A.5: Caso de uso Amosar mapa edificio.

IDENTIFICADOR	Cambiar piso
Nome	Cambiar piso
Descripción	Amosa o mapa dun piso diferente ao amosado dentro dun edificio
Actor	Usuario Android
Precondicións	Amosar o mapa dun edificio e que teña máis dun nivel
Poscondicións	Cambiará a imaxe do edificio e porá o novo nivel
Escenario	Na pantalla principal, pulsar sobre o botón do edificio que se desexa amosar na zona dereita da pantalla ou cambiar fisicamente de piso

Táboa A.6: Caso de uso Cambiar piso.

IDENTIFICADOR	Localización no interior dun edificio
Nome	Localización no interior dun edificio
Descripción	Identifica a localización dun usuario dentro dun edificio
Actor	Usuario Android
Precondicións	Debe ter permisos para amosar o mapa edificio e atoparse fisicamente dentro del
Poscondicións	Amosará un ícono no mapa representando a localización do dispositivo móvil
Escenario	Debemos ter aberta a pantalla principal e atoparnos dentro do edificio en cuestión

Táboa A.7: Caso de uso Localización no interior dun edificio.

IDENTIFICADOR	Amosar lista de POIs dun edificio
Nome	Amosar lista de POIs dun edificio
Descripción	Amosa un spinner con todos os POIs dun edificio
Actor	Usuario Android
Precondicións	Debe ter cargados os POIs dun edificio
Poscondicións	Poderá amosar un spinner con todos os POIs dun edificio
Escenario	Na pantalla principal, pinchar sobre o botón de amosar POIs

Táboa A.8: Caso de uso Amosar lista de POIs dun edificio.

IDENTIFICADOR	Amosar POI no mapa
Nome	Amosar POI no mapa
Descripción	Amosa a localización dun POI do edificio no mapa
Actor	Usuario Android
Precondicións	Debe ter cargados os POIs dun edificio
Poscondicións	Amósase unha marca dentro do mapa na súa localización
Escenario	Na pantalla principal, seleccionar un POI no spinner coa lista de POIs

Táboa A.9: Caso de uso Amosar POI no mapa.

IDENTIFICADOR	Amosar información dun POI
Nome	Amosar información dun POI
Descripción	Amosa toda a información dun POI
Actor	Usuario Android
Precondicións	Debe ter seleccionado un POI dentro do spinner de POIs dun edificio
Poscondicións	Poderá amosar toda a información do POI en cuestión: nome, descripción, tempo de visita...
Escenario	Na pantalla principal, seleccionar un POI concreto no spinner de POIs ou dentro do mapa e premer o botón de detalles

Táboa A.10: Caso de uso Amosar información dun POI.

IDENTIFICADOR	Guiar a POI
Nome	Guiar a POI
Descripción	Amosa a ruta que ten que realizar o usuario para chegar ao POI debuxado no mapa do edificio
Actor	Usuario Android
Precondicións	Seleccionar un POI dun edificio amosado no mapa
Poscondicións	Amosará a localización actual, o POI ao que desexa ir e unha ruta que une ambos
Escenario	Na pantalla principal, pinchar sobre o marcador dun POI e despois sobre a información que amosa

Táboa A.11: Caso de uso Guiar a POI.

IDENTIFICADOR	Amosar lista de percorridos dun edificio
Nome	Amosar lista de percorridos dun edificio
Descripción	Amosa un spinner con todos os percorridos dun edificio
Actor	Usuario Android
Precondicións	Debe ter cargados os percorridos dun edificio
Poscondicións	Poderá amosar un spinner con todos os percorridos dun edificio
Escenario	Na pantalla principal, pinchar sobre o botón de amosar percorridos

Táboa A.12: Caso de uso Amosar lista de percorridos dun edificio.

IDENTIFICADOR	Amosar percorrido no mapa
Nome	Amosar percorrido no mapa
Descripción	Amosa todos os POIs dentro dun percorrido e indica a orde mediante liñas sinalizadas
Actor	Usuario Android
Precondicións	Debe ter cargados os percorridos dun edificio
Poscondicións	Amósanse as marcas dos POIs e unha frecha entre cada un deles indicando a dirección de visita
Escenario	Na pantalla principal, seleccionar un percorrido no spinner coa lista de percorridos

Táboa A.13: Caso de uso Amosar percorrido no mapa.

IDENTIFICADOR	Guiar a través de percorrido
Nome	Guiar a través de percorrido
Descripción	Amosa as rutas que ten que realizar o usuario para viaxar entre os POIs dun percorrido no mapa do edificio
Actor	Usuario Android
Precondicións	Seleccionar un percorrido dun edificio amosado no mapa
Poscondicións	Amosará a localización actual, os POIs do percorrido polos que xa se pasou e os que faltan por pasar. Os POIs estarán unidos por liñas de distintas cores indicando o seu estado
Escenario	Na pantalla principal, pinchar sobre o marcador dun POI e despois sobre a información que amosa

Táboa A.14: Caso de uso Guiar a través de percorrido.

IDENTIFICADOR	Amosar información dun percorrido
Nome	Amosar información dun percorrido
Descripción	Amosa toda a información dun percorrido
Actor	Usuario Android
Precondicións	Debe ter seleccionado un percorrido no spinner de percorridos
Poscondicións	Poderá amosar toda a información do percorrido en cuestión: nome, descripción, tempo de visita...
Escenario	Na pantalla principal, seleccionar un percorrido concreto no spinner de percorridos e pinchar no botón de detalles

Táboa A.15: Caso de uso Amosar información dun percorrido.

IDENTIFICADOR	Amosar lista de imaxes dun POI
Nome	Amosar lista de imaxes dun POI
Descripción	Amosa a lista de imaxes dun POI
Actor	Usuario Android
Precondicións	Debe ter seleccionado un POI que teña imaxes
Poscondicións	Amosarase unha lista con todas as imaxes dispoñíbeis do POI
Escenario	Na pantalla de detalle dun POI, premer no botón de visualizar lista de imaxes

Táboa A.16: Caso de uso Amosar lista de imaxes dun POI.

IDENTIFICADOR	Amosar imaxe dun POI
Nome	Amosar imaxe dun POI
Descripción	Amosa unha imaxe dun POI
Actor	Usuario Android
Precondicións	Debe ter seleccionado un POI que teña imaxes
Poscondicións	Amosarase unha pantalla coa imaxe escollida
Escenario	Na pantalla de detalle dun POI, premer no botón de visualizar imaxe e escoller unha entre as dispoñíbeis

Táboa A.17: Caso de uso Amosar imaxe dun POI.

IDENTIFICADOR	Crear POI
Nome	Crear POI
Descripción	Crea un POI dentro dun edificio, asignándolle unha posición concreta e outros datos de interese
Actor	Xestor de contido
Precondicións	Debe seleccionarse un edificio e ter permisos de administración sobre el
Poscondicións	Haberá un novo POI asociado a ese edificio
Escenario	Debemos entrar no modo de edición sobre un edificio. Seleccionar o botón de engadir POI e pulsar sobre a localización desexada no mapa. Inserir a información sobre o POI e pulsar no botón Gardar.

Táboa A.18: Caso de uso Crear POI.

IDENTIFICADOR	Modificar POI
Nome	Modificar POI
Descripción	Modifica a información dun POI
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio. Debe seleccionarse un POI e visualizar a súa información
Poscondicións	O POI terá nova información
Escenario	Debemos entrar no modo de edición sobre un edificio. Visualizar a información dun POI e modificar os datos desexados. Pulsar no botón Gardar

Táboa A.19: Caso de uso Modificar POI.

IDENTIFICADOR	Eliminar POI
Nome	Eliminar POI
Descripción	Eliminar un POI dun edificio
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio. Debe seleccionarse un POI e visualizar a súa información
Poscondicións	O POI desaparecerá do edificio
Escenario	Debemos entrar no modo de edición sobre un edificio. Visualizar a información e pulsar no botón Eliminar

Táboa A.20: Caso de uso Eliminar POI.

IDENTIFICADOR	Estimar tempo POI
Nome	Estimar tempo POI
Descripción	Estima e asocia o tempo adicado a un POI
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio. Debe seleccionarse un POI e visualizar a súa información
Poscondicións	O POI verá actualizada a súa información
Escenario	Debemos entrar no modo de edición sobre un edificio. Visualizar a información, engadir a estimación e pulsar no botón Gardar

Táboa A.21: Caso de uso Estimar tempo POI.

IDENTIFICADOR	Engadir imaxe
Nome	Engadir imaxe a POI
Descripción	Engade una nova imaxe a un POI
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio. Debe seleccionarse un POI e visualizar a súa información
Poscondicións	Haberá unha nova imaxe asociada a ese POI
Escenario	Debemos entrar no modo de edición sobre un edificio. Visualizar a información dun POI e pulsar sobre o botón de engadir Imaxe. Inserir os datos da imaxe e seleccionala. Pulsar sobre Gardar

Táboa A.22: Caso de uso Engadir imaxe.

IDENTIFICADOR	Modificar datos imaxe
Nome	Modificar datos imaxe
Descripción	Modifica a información da imaxe dun POI
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio. Debe seleccionarse un POI e visualizar a súa información. Posteriormente, seleccionar unha imaxe
Poscondicións	A imaxe terá nova información
Escenario	Debemos entrar no modo de edición sobre un edificio. Visualizar a información dun POI e acceder a unha imaxe. Modificar os seus datos desexados e pulsar no botón Gardar

Táboa A.23: Caso de uso Modificar datos imaxe.

IDENTIFICADOR	Eliminar imaxe
Nome	Eliminar imaxe
Descripción	Eliminar imaxe
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio. Debe seleccionarse un POI e visualizar a súa información. Posteriormente, seleccionar unha imaxe
Poscondicións	A imaxe desaparecerá do edificio
Escenario	Debemos entrar no modo de edición sobre un edificio. Visualizar a información dun POI e acceder a unha imaxe. Posteriormente, pulsar no botón Eliminar

Táboa A.24: Caso de uso Eliminar imaxe.

IDENTIFICADOR	Crear percorrido
Nome	Crear percorrido
Descripción	Seleccionar unha lista de POIs dentro dun edificio e ordenalos para crear un percorrido. Tamén se debe proporcionar información referente ao percorrido
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio en cuestión e que existan más de dous POIs
Poscondicións	Ordenará unha serie de POIs dun edificio que poderá seguir un usuario
Escenario	Debemos entrar no modo de edición sobre un edificio. Pulsar sobre o botón de crear percorrido. A continuación, seleccionar os POIs na orde desexada. Pulsar sobre o botón de aceptar ao finalizar. Inserir os datos do percorrido. Pulsar sobre Gardar

Táboa A.25: Caso de uso Crear percorrido.

IDENTIFICADOR	Modificar información percorrido
Nome	Modificar información percorrido
Descripción	Modifica a información referente a un percorrido
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio en cuestión e recuperar a información dun percorrido
Poscondicións	O percorrido gardará as novas condicións
Escenario	Debemos entrar no modo de edición sobre un edificio. Seleccionar un percorrido e visualizar os seus datos. Modificar os datos desexados e pulsar sobre Gardar

Táboa A.26: Caso de uso Modificar información percorrido.

IDENTIFICADOR	Eliminar percorrido
Nome	Eliminar percorrido
Descripción	Elimina un percorrido do sistema
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio en cuestión e recuperar a información dun percorrido
Poscondicións	O percorrido deixará de existir
Escenario	Debemos entrar no modo de edición sobre un edificio. Seleccionar un percorrido e visualizar os seus datos. Pulsar sobre Eliminar

Táboa A.27: Caso de uso Eliminar percorrido.

IDENTIFICADOR	Estimar tempo percorrido
Nome	Estimar tempo percorrido
Descripción	Estima e asocia o tempo adicado a un percorrido
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio. Debe seleccionarse un percorrido e visualizar a súa información
Poscondicións	O percorrido verá actualizada a súa información
Escenario	Debemos entrar no modo de edición sobre un edificio. Visualizar a información, engadir a estimación e pulsar no botón Gardar

Táboa A.28: Caso de uso Estimar tempo percorrido.

IDENTIFICADOR	Engadir POI a percorrido
Nome	Engadir POI a percorrido
Descripción	Engade un POI a un percorrido xa existente
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio en cuestión e que o POI que se quere engadir non estea no percorrido seleccionado
Poscondicións	Introducirá o POI ao inicio do percorrido, ao final ou entre dous POIs concretos
Escenario	Debemos entrar no modo de edición sobre un edificio e seleccionar un percorrido. Se desexamos inserir o novo POI ao inicio ou ao final do percorrido, debemos seleccionalo directamente. Se desexamos introducilo entre dous POIs, debemos seleccionar a liña que os une primeiro. A continuación, pulsar sobre o botón de aceptar

Táboa A.29: Caso de uso Engadir POI a percorrido.

IDENTIFICADOR	Eliminar POI de percorrido
Nome	Eliminar POI de percorrido
Descripción	Elimina un POI de dentro dun percorrido
Actor	Xestor de contido
Precondicións	Debe ter permisos de administración sobre o edificio en cuestión e seleccionar un percorrido. Deben existir más de tres POIs
Poscondicións	O POI deixará de existir dentro do percorrido
Escenario	Debemos entrar no modo de edición sobre un edificio. Seleccionar un percorrido e pulsar sobre o POI en cuestión. Pulsar sobre o botón de aceptar

Táboa A.30: Caso de uso Eliminar POI de percorrido.

IDENTIFICADOR	Dar de alta un edificio
Nome	Dar de alta un edificio
Descripción	Engade un novo edificio á nosa plataforma
Actor	Administrador do sistema
Precondicións	O edificio a engadir non debe existir na nosa plataforma e si debe estar incluído en Situm
Poscondicións	O edificio estará dispoñible na nosa plataforma
Escenario	Débese coñecer o identificador en Situm do edificio e inserir os seus datos a través dun Sistema Xestor de Base de Datos

Táboa A.31: Caso de uso Dar de alta un edificio.

IDENTIFICADOR	Dar permiso a un usuario sobre un edificio
Nome	Dar permiso a un usuario sobre un edificio
Descripción	Proporciona a un usuario a posibilidade de creación, modificación e eliminación de contido sobre un edificio concreto
Actor	Administrador do sistema
Precondicións	O edificio debe existir na nosa base de datos e o usuario ao que se lle quere dar o permiso debeu autenticarse algunha vez na aplicación
Poscondicións	O usuario poderá modificar o contido dese edificio
Escenario	Débense coñecer os identificadores do edificio e do usuario para asocialos a través dun Sistema Xestor de Base de Datos

Táboa A.32: Caso de uso Dar permiso a un usuario sobre un edificio.

IDENTIFICADOR	Dar de alta unha conta Situm
Nome	Dar de alta unha conta Situm
Descripción	Engade unha nova conta de Situm ao noso sistema
Actor	Administrador do sistema
Precondicións	A conta non debe existir na nosa base de datos e si no sistema de Situm
Poscondicións	A conta estará accesible no noso sistema
Escenario	Débense coñecer os datos da conta de Situm e inserir os seus datos a través dun Sistema Xestor de Base de Datos

Táboa A.33: Caso de uso Dar de alta unha conta Situm.

IDENTIFICADOR	Asignar conta Situm a Usuario
Nome	Asignar conta Situm a Usuario
Descripción	Proporciona a un usuario a posibilidade de utilización dunha conta de Situm para acceder ao sistema, podendo visualizar os edificios asociados a esa conta
Actor	Administrador do sistema
Precondicións	A conta de Situm debe existir na nosa base de datos e o usuario ao que se lle quere asignar debeu autenticarse algunha vez na aplicación
Poscondicións	O usuario podrá acceder ao sistema con esa conta
Escenario	Débense coñecer os identificadores da conta e do usuario para asocialos a través dun Sistema Xestor de Base de Datos

Táboa A.34: Caso de uso Dar permiso a un usuario sobre un edificio.

Apéndice B

Instalación do sistema Caronte

Neste apéndice detallaranse as principais fases para a instalación e posta a punto do noso sistema Caronte. En primeiro lugar, indicaranse os requisitos do sistema. A continuación, os pasos para a súa instalación e despregue, tanto do servidor como da aplicación Android. Por último, remataremos co manual do usuario e unha guía rápida de funcionamento.

B.1. Requisitos

B.1.1. Servidor Caronte

B.1.2. Aplicación Android: Caronte

Para poder instalar e utilizar a aplicación, requírese un dispositivo móvil que cumpla unha serie de requisitos software, pero sobre todo hardware. A lista de necesidades pódese ver a continuación:

- Sistema operativo Android superior ou igual á versión 4.4 (Kit Kat).
- Sensor GPS.
- Conexión Wi-Fi.
- Conexión datos.

- Bluetooth (recomendado).
- 1,5 GB de memoria RAM para un uso .
- 7 MB libres de espazo no dispositivo. Máis se se desexan visualizar imaxes de puntos de interese.

B.2. Instalación

B.2.1. Servidor Caronte

B.2.2. Aplicación Android: Caronte

A instalación da aplicación pode levarse a cabo a través da tenda oficial de Google: Google Play. Non obstante, tamén se pode proceder á instalación manual do arquivo APK se se habilita a instalación de aplicacións con orixe descoñecida nos axustes de seguridade do teléfono.

Non hai ningún tipo de necesidade a maiores no momento da instalación, xa que a petición de permisos necesarios para o funcionamento da aplicación solicítanse dinamicamente na súa execución.

B.3. Manuales de usuario

Nesta sección comentaranse os manuais de usuario para os tres actores do sistema Caronte: administrador, xestor de contidos e usuario anónimo.

B.3.1. Administrador do servidor Caronte

despliegue, inicialización
cambio de rol

B.3.2. Aplicación Caronte para usuarios anónimos

vistas e guía rápida de funcionamento

B.3.3. Aplicación Caronte para xestores de contidos

usuario autenticado

usuario con permisos de edición: POI, recorridos, etc.

Apéndice C

Localización en interiores de Situm

Neste apéndice trataranse os pasos precisos para realizar a configuración de Situm que é necesaria para a utilización do sistema de Situm da nosa aplicación.

C.1. Nova conta en Situm

O primeiro paso para a configuración e calibración dun edificio é a creación dunha conta de usuario na páxina de Situm. Para a súa creación só é preciso un correo electrónico e un contrasinal.

C.2. Dashboard de Situm

Unha vez creada a conta de usuario en Situm permitirase o acceso ao seu Dashboard (ver figura C.1). Dende esta páxina web poderemos crear edificios, cargar os seus mapas, establecer grafos e incluso observar todos os usuarios localizados dentro dos nosos edificios.

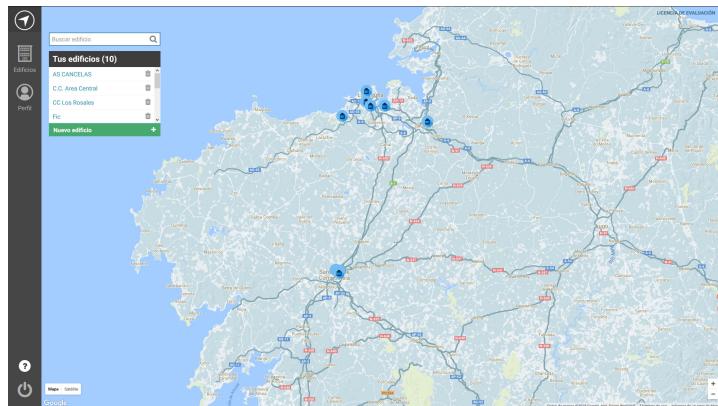


Figura C.1: Vista inicial do dashboard de Situm.

C.3. Editar edificios en Situm

O dashboard de Situm permítenos a creación e edición de edificios. Ao abrilo veremos un mapa onde visualizaremos as posicións dos nosos edificios e unha lista á esquerda cunha relación dos mesmos. Dende aquí poderemos modificar os edificios xa existentes ou crear outros novos. Para a creación dun novo edificio deberemos indicar a información do mesmo, coma o seu nome e unha descripción. No segundo paso debemos crear as súas plantas, para o cal precisaremos os seus planos. Para cada unha das plantas debemos situar o seu plano no mapa para que coincida co seu emprazamento real, podendo modificar a súa escala e rotalo. Este paso é básico para que funcione correctamente a localización no interior do edificio, polo que se debe axustar ao máximo os planos. Na figura C.2 pódese observar a vista de modificación dun edificio.

C.4. Calibración de cada planta

Para a calibración das plantas dos nosos edificios precisamos a aplicación de calibrado proporcionada por Situm: Situm Mapping Tool (ver figura C.3). É unha aplicación moi simple na que debemos indicar a nosa situación no mapa mentres nos movemos polo edificio. O obxectivo é cubrir a superficie de todas as plantas por onde poden pasar os usuarios. O proceso de configuración é o seguinte: partindo dunha posición inicial, debemos indicar na aplicación a nosa situación. Posteriormente haberá que camiñar en

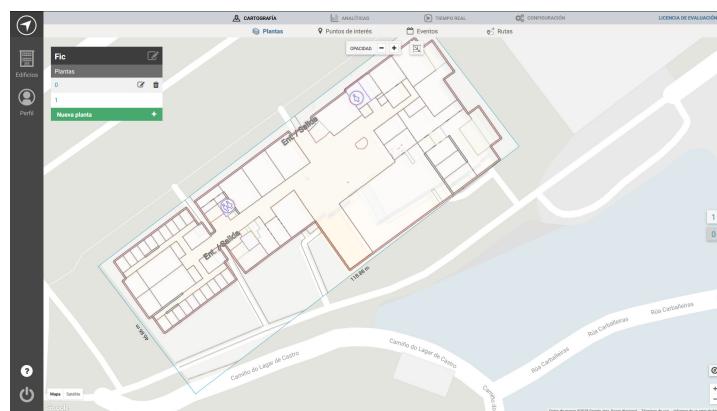


Figura C.2: Vista de modificación dun edificio.

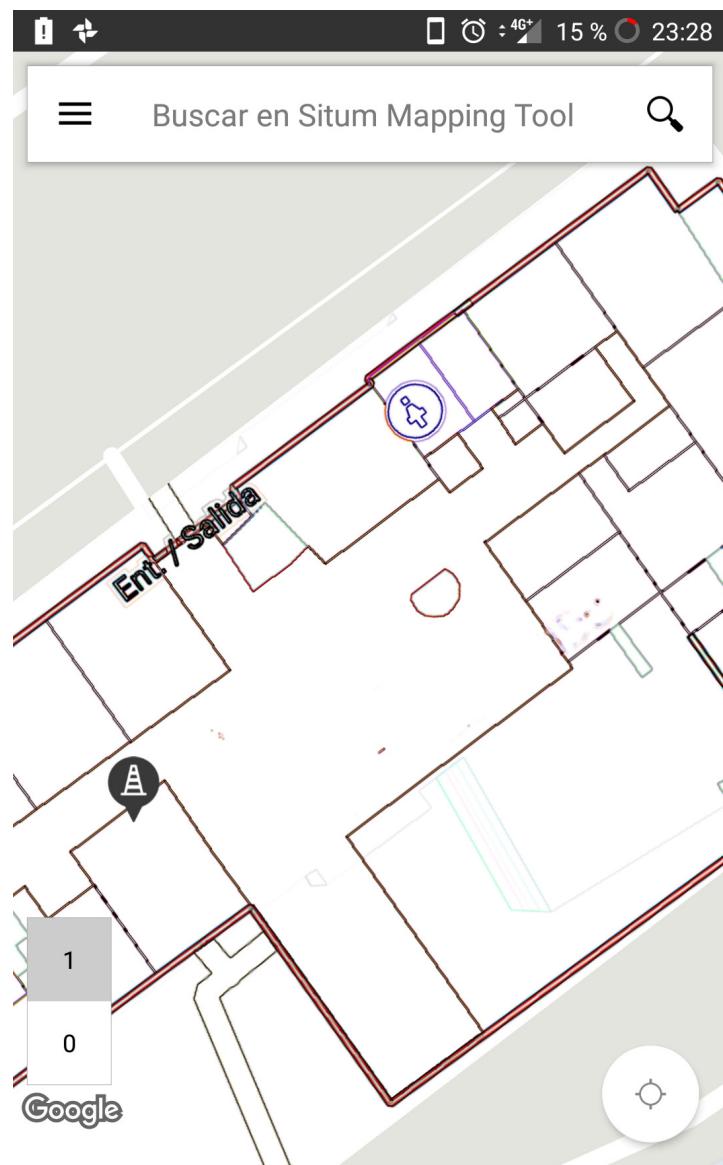
liña o máis recta posíbel mentres indicamos cada poucos metros onde nos atopamos. Cando desexemos rematar con esa calibración temos que parala e enviala aos servidores de Situm. Haberá que repetir este proceso varias veces ata cubrir a superficie do edificio. É recomendábel facer recalibracións periódicas dos edificios para non perder precisión.

C.5. Punto de interese Situm

Situm dá a posibilidade da creación de puntos de interese propios para a realización de accións concretas. No noso sistema non facemos uso deles pois temos os nosos propios puntos de interese pero explícase o proceso de creación pois é unha das opcións máis útiles do seu sistema. Dentro do dashboard de Situm e despois de seleccionar o edificio no que queremos crear os puntos de interese, escolleremos a pestana chamada "Puntos de interese". Chegado a este punto visualizaranse os mapas do edificio xunto cos puntos de interese xa creados, dispoñíbeis para a súa edición ou eliminación. Por suposto, tamén está dispoñíbel a creación de novos puntos, que poderemos situar directamente sobre o mapa e para os cales se dará algún tipo de información. Este proceso pode observarse na captura [C.4](#)

C.6. Definición de rutas

No Dashboard hai unha sección dende a cal poderemos definir as rutas dentro do edificio. Dende ela pódense unir os distintos grafos creados por Situm na calibración do edificio para indicar as posíbeis rutas que poderán seguir os usuarios. Non hai máis que pulsar os grafos que queremos combinar para crear unha ruta. Tamén se permite a unión de puntos entre pisos distintos, facendo click sobre un dos puntos que se desexa unir para despois seleccionar o outro punto e marcar a opción. A captura [C.5](#) mostra a ventá de definición de rutas.



Fic



Figura C.3: Aplicación de calibración: Situm Mapping Tool.

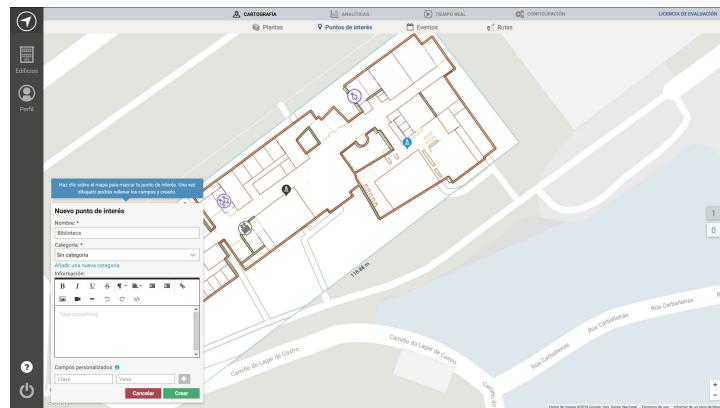


Figura C.4: Proceso de creación de POIs en Situm.

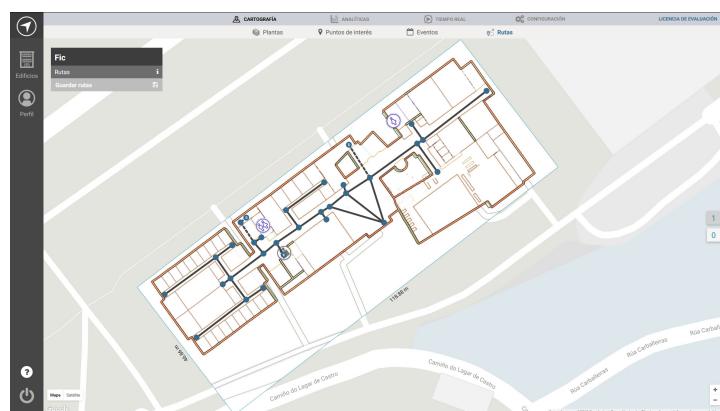


Figura C.5: Proceso de creación de rutas.

Apéndice D

Contido do DVD adxunto

