Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DO SOFTWARE

Aplicación móbil multiplataforma para proporcionar información de accesibilidade a persoas con diversidade funcional física

Estudante: Alejandro González Martínez
Dirección: Fernando Bellas Permuy

A Coruña, setembro de 2019.

Aos que hoxe non están aquí para ler estas liñas pero devecerían por facelo.

Agradecementos

En primeiro lugar quero darlle as grazas a miña familia, porque sen eles hoxe non estaría aquí escribindo estas liñas. Especial mención á meu avó, que de estares aquí lería cada unha das liñas deste documento aínda sen entender nin unha soa palabra, só por orgullo.

A todos os meus amigos, dende aqueles que me levan aguantado dende que teño uso de razón ata os que apareceron nas incontables horas de facultade para agora quedarse.

A Fernando, polas reunións no despacho que sempre traían algo bó consigo e que aportaban algo de orde no meu labirinto de dúbidas.

Resumo

Neste traballo fin de grado preséntase unha aplicación móbil multi-plataforma (iOS e Android) orientada a facilitar o día a día as persoas con diversidade funcional física. A finalidade principal da aplicación é a de ofrecer ao usuario un filtro a través do cal poida localizar calquera tipo de lugar nunha cidade (bares,parques...) en función das necesidades de accesibilidade que especifique (ascensor, ramplas de acceso...) e da localización desexada.

A súa vez, cos lugares obtidos do filtrado pódese interactuar de diferentes maneiras, como pode ser: dando unha opinión do lugar, puntuando a súa accesibilidade, etc.

O núcleo da aplicación é unha API Rest implementada mediante o framework SpringBoot e persistindo a información nunha base de datos PostgreSQL. Esta API é consumida por unha aplicación móbil implementada utilizando JavaScript con axuda de React Native como framework.

Abstract

In this final degree project a multi-platform mobile application (iOS and Android) is presented aimed to ease the day-to-day life of people with physical functional diversity. The main purpose of the application is to offer to the user a filter through which he can locate any type of place in a city (pubs,parks ...) depending on the accessibility needs you specify (elevator, access ramps. ..) and the desired location.

In turn, with the places obtained from the filtering you can interact in different ways, such as: giving an opinion of the place, rating its accessibility, etc.

The core of the application is a Rest API implemented using the SpringBoot framework and persisting the information in a PostgreSQL database. This API is consumed by a mobile application implemented using JavaScript with the help of React Native as a framework.

Palabras chave:

- Diversidade funcional física
- API REST
- Spring Boot
- JavaScript
- React Native
- Redux
- Scrum
- PostgreSQL

Keywords:

- Diverse physical functionality

Índice Xeral

Índice de Figuras

Índice de Táboas

Capítulo 1

Introdución

1.1 Contexto

Adía de hoxe e despois de anos de loita e reivindicación, as persoas con diversidade funcional estanse facendo o oco que lles pertence na sociedade. Parte desta reinvindicación ven dada polo feito de que a este tipo de persoas anos atrás simplemente se lles ignoraba ou se menosprezaban, o que se resumía nun mundo deseñado por e para as persoas que non sufrían ningún tipo de discapacidade, o que provocaba unha limitación aínda máis grande da que estas persoas xa sufrían.

AccessibilitApp (nome que se lle outorgou a aplicación aquí explicada) nace a raíz desta loita para proporcionar a todas estas persoas unha pequena guía que lles permita vivir a súa vida dunha maneira máis cómoda e evitar así todas aquelas trabas que poidan surxir no seu día a día.

1.2 Obxectivos

O obxectivo deste proxecto é crear unha plataforma que permita as persoas con diversidade funcional física obter información sobre a accesibilidade de todo tipo de lugares como poden ser: bares, parques, restaurantes...

Aínda que vai destinado a axudar a este tipo de persoas, calquera individuo pode rexistrarse na aplicación e interacturar co sistema, axudando desta maneira a facer crecer a plataforma. A hora de interactuar co sistema os usuarios poden realizar multitude de accións como poden ser: a búsqueda de lugares utilizando diferentes filtros, puntuar e comentar un lugar en concreto ou marcar certa localización como favorita entre outras funcionalidades.

Ademais, todo aquel usuario que posúa unha conta no sistema poderá participar no crecemento do mesmo. Para isto, permíteselle aos usuarios realizar peticións para engadir novos lugares, modificar lugares xa existentes ou eliminalos.

Outro dos obxectivos que se pretende alcanzar é que a aplicación ademais de ser funcional e útil, dispoña dunha interface de usuario que sexa cómoda e atractiva para o mesmo dado que a usabilidade é un aspecto moi importante.

Por último, búscase aproveitar a realización deste proxecto para adquirir coñecementos naquelas tecnoloxías que nunca antes se trataran, como poden ser React Native, Javascript, Spring Boot... E desta maneira aprender, pois é o obxectivo principal da carreira.

1.3 Visión global do sistema

O sistema consistirá nunha aplicacion móbil multiplataforma implementada empregando JavaScript como linguaxe xunto con React Native [1] como librería/Framework. A través desta aplicación o usuario interactuará directamente podendo acceder as funcionalidades que proporciona o sistema.

Para o seu funcionamento, esta aplicación fai uso dunha API Rest implementada mediante Spring Boot [2], na que radicará toda a lóxica de negocio do sistema.

Para finalizar, e en referencia a persistencia dos datos que utiliza o servizo Rest, cabe indicar que se utiliza PostgreSQL como xestor de base de datos, sendo esta unha base de datos relacional SQL.

Na Figura 1.1 pode observarse un esquema xeral da arquitectura do sistema.

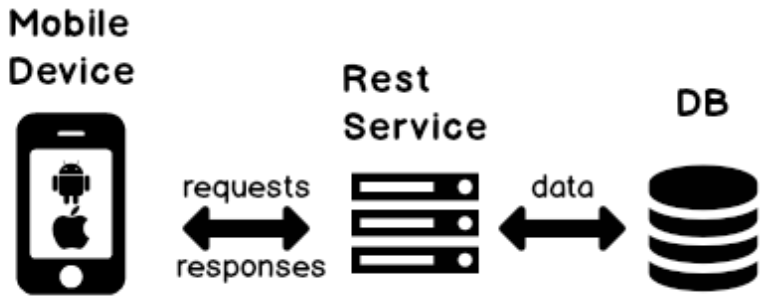

Figura 1.1: Arquitectura do sistema

Capítulo 2

Estado del arte

Ahora de comezar coa posta en marcha do proxecto e como toma de referencias fíxose un estudo exhaustivo das aplicacións existentes que trataban a temática que se considera neste traballo, analizando non só as que o fan directamente senón tamén aquelas que o fan de maneira indirecta ou levemente.

As aplicacións que cumpren estas condicións e que foron probadas e examinadas son as seguintes:

2.1 DisCar

Aplicación orientada a localización de prazas de aparcamento para persoas con diversidade funcional de tipo físico. Está feita por un desenrolador sen ningún tipo de relación con ningunha empresa nin organización. A aplicación en si intenta paliar ou reducir un problema que ten relación directa coa aplicación aquí tratada, pero sen ofrecer o mesmo tipo de funcionalidade (simplemente indica as prazas de aparcamento coa identificación de minusvalía).

2.2 Iello

Aplicación destinada a procura de aparcamento para persoas con diversidade funcional física. Realiza a mesma funcionalidade que a aplicación anteriormente analizada, pero ao contrario que a anterior esta funciona de maneira híbrida (é dicir, feita con tecnoloxías web e correndo dentro dun navegador embebido). Pola información proporcionada pola Play Store pódese saber que esta aplicación foi realizada por unha universidade Italiana como un traballo universitario (caso similar ao actual).

2.3 Accessibility Plus

Esta aplicación permite localizar e solicitar taxis adaptados ao transporte de persoas con diversidade funcional física. Ademais desta funcionalidade principal tamén permite consultar certos puntos que posúen unha boa accesibilidade sendo estes introducidos pola propia aplicación. Ao contrario que as anteriores esta aplicación si está realizada por unha empresa ou organización, xa que a autoría da mesma pertence a Fundación Vodafone.

2.4 EsAccesible App

A seguinte Aplicación permite valorar a accesibilidade física dun lugar concreto. Trátase dunha aplicación híbrida. A pesares de coincidir na funcionalidade básica de valorar a accesibilidade física dun lugar, esta aplicación carece de calquera tipo de reseña sobre os lugares ou un filtro de búsqueda entre outras moitas funcionalidades.

Capítulo 3

Metodoloxía

No ámbito software, unha metodoloxía é un marco de traballo que impón un proceso disciplinado sobre o desenrolo do software co fin de facelo máis predicible e eficiente. O seu obxectivo principal é aumentar a calidade do produto en todas e cada unha das fases do desenrolo.

Para a consecución do proxecto utilizarase como método de traballo unha metodoloxía áxil de ciclo de vida incremental. As metodoloxías áxiles son aquelas que permiten adaptar a forma de traballo as condicións do proxecto, conseguindo flexibilidade e inmediatez na resposta para amoldar o proxecto e o seu desenrolo as circunstancias específicas do entorno.

Entre os moitos beneficios que aporta este tipo de metodoloxías, os máis destacados son:

- Melloran a produtividade do equipo de traballo.
- Melloran a calidade dos produtos finais.
- Facilitan os cambios durante o proceso.
- Seguen un proceso iterativo e incremental, polo que ao final de cada iteración obtense software funcional.
- Aumentan a participación do cliente no proceso evitando gran cantidade de problemas e mellorando a súa satisfacción.
- Redúcense os riscos debido a existencia dun software funcional en cada iteración.

Non se escolleu ningunha metodoloxía áxil concreta. Realízase un desenrolo iterativo da aplicación e o servizo. En cada iteración (na que se fai: análise, deseño, implementación e probas), incorpórase un conxunto de funcionalidades (casos de uso) tanto ó servizo como á aplicación móbil, de xeito que cada iteración produza un sistema que se pode usar (aínda que con funcionalidade limitada, excepto na última iteración).

A pesar de non escoller ningunha metodoloxía concreta, a que máis se lle podería aproximar sería Scrum [3], xa que durante o desenvolvemento do proxecto utilizáronse certos artefactos e definicións propias de Scrum que se explicarán a continuación. O feito de non chamarlle Scrum a metodoloxía utilizada é polo simple feito de que en Scrum existe un equipo con certos papeis ben definidos que posúen persoas diferentes (Scrum Master, Product owner...) e neste caso concreto o equipo está formado por unha soa persoa, polo que, sendo estritos, non se lle podería chamar Scrum.

3.1 Sprints

Un Sprint é unha sección de tempo cunha duración recomendada de entre unha e catro semanas na cal se desenrolan parte dos requisitos definidos do proxecto.

En cada unha destas iteracións selecciónanse con axuda do cliente unha serie de funcionalidades a realizar, obtendo así ao final da iteración un produto funcional que o satisfae.

Durante o Sprint, realízanse as tarefas de análise, deseño, implementación e probas, xerando desta maneira unha serie de artefactos que se explicarán a continuación.

3.2 Artefactos

No marco de traballo Scrum e mediante a súa aplicación xéranse unha serie de elementos físicos que garanten a transparencia do proceso e facilitan desta maneira a organización do proxecto e máis a toma de decisións. Estes elementos son os chamados artefactos. Neste caso concreto e aínda que non se trate da aplicación estrita de Scrum, tamén se utilizarán estes artefactos coas mesmas definicións e normas.

Neste caso concreto a xestión dalgúns artefactos como son o Product Backlog ou o Sprint Backlog foi levada a cabo coa ferramenta de xestión de proxectos Redmine. En cada un dos sprints creouse unha ”Versión” do proxecto correspondendo coa iteración en tránsito, engadindo aquelas tarefas que correspondían as do Sprint Backlog. Para cada unha destas tarefas engadíanse certos datos como podían ser unha descrición, unha planificación temporal, o tipo de tarefa ou o estado no que se atopaba a mesma, permitindo desta maneira levar un seguimento exhaustivo do Sprint.

3.2.1 Product Backlog

O Product Backlog é un listado ordenado que recolle todas as necesidades e requisitos que resultan de interese para o cliente ou os usuarios finais. Trátase dunha lista dinámica que pode verse alterada a medida que o proxecto o requira, tanto engadindo novas entradas como modificando outras xa existentes.

Estas necesidades aparecen descritas en forma de historias de usuario, cunha linguaxe comprensible para calquera, e ordenadas en función do nivel de detalle que posúan. O Propietario do Produto ou Product Owner (papel que neste caso realizaría a única persoa do equipo) é o encargado de manter a lista actualizada, asignando prioridades e reordenándoa ao comezo de cada sprint.

3.2.2 Sprint Backlog

O Sprint Backlog é o conxunto dos elementos tomados do Product Backlog para ser realizados durante o Sprint. As tarefas contidas no Sprint Backlog son administradas polo propio equipo de desenvolvemento, que se auto-organiza para asumir o traballo da pila do Sprint (Sprint Backlog), tanto durante a planificación do Sprint como ao largo do mesmo.

3.2.3 Historias de usuario

As historias de usuario son a técnica utilizada polas metodoloxías áxiles para a especificación de requisitos. Son unha forma rápida de administrar os requisitos dos usuarios de maneira sinxela e evitando documentación tediosa e sobrecustos de tempo.

A niveis prácticos, unha historia de usuario é unha representación dun requisito escrito nunha ou dúas frases utilizando a linguaxe común do usuario. Grazas a isto e debido a súa natureza, permiten responder rapidamente aos requisitos cambiantes.

3.3 Roles

Os equipos en Scrum son auto-organizados, multifuncionais e definen tres roles claramente diferenciados: o Propietario do Produto (Product Owner), o Equipo de Desenvolvemento (Development Team) e o Scrum Master.

Tendo en conta que este proxecto é un Traballo de Fin de Grado e que está realizado por unha única persoa (o alumno), tanto o equipo de desenvolvemento como os roles de Product Owner e Scrum Master serán realizados pola mesma persoa, o alumno.

Sen embargo, o director do proxecto tamén participou como parte no rol de Scrum Master, axudando á resolución dos problemas e impedimentos atopados durante o desenvolvemento (tarefa do Scrum Master cara o equipo de desenvolvemento).

3.3.1 Product Owner

É o responsable de maximizar o valor do produto e o traballo do Equipo de Desenvolvemento (Development Team). Ademais, o Propietario do Produto (Product Owner) é a única

persoa responsable de xestionar a Pila do Produto (Product Backlog), podendo verse aquí reflexadas as súas decisións.

3.3.2 Scrum Master

O Scrum Master é o responsable de asegurar que se entenda e se adopte Scrum. Os Scrum Masters fan isto asegurándose de que o Equipo Scrum traballa axustándose á teoría, prácticas e regras de Scrum.

3.3.3 Development Team

Componse en base aos profesionais que realizan o traballo de entregar un incremento do produto "Terminado" (atópase en condicións de ser utilizado e cumpre a definición de "Terminado" do equipo Scrum) que potencialmente se poida por en produción ao final de cada Sprint. Só os membros do Equipo de desenvolvemento (Development Team) participan na creación do Incremento.

3.4 Reunións

O éxito de calquera proxecto pasa por atender un elemento en concreto: a comunicación. É necesario conseguir que a comunicación flúa de maneira natural entre os tres roles e as mensaxes sexan efectivas. Isto conséguese a través das reunións de equipo, as cales deben atender a necesidades concretas.

Neste caso concreto realizouse en primeiro lugar unha reunión inicial entre o alumno e o profesor para decidir os requisitos que se ían cubrir co Product Backlog e de que maneira se ían organizar os Sprints.

Debido a que o equipo de desenvolvemento está formado por unha única persoa (o alumno), decidiuse prescindir das reunións diarias, xa que estas buscan a posta en común do estado do traballo entre os distintos membros do equipo de desenvolvemento (Development Team).

Coa realización dos sucesivos sprints, fóronse levando a cabo para cada un deles as reunións de planificación (Sprint Planning), revisión (Sprint Review) e retrospectiva (Sprint Retrospective), participando de novo nelas o alumno e máis o director. Nestas reunións fóronse decidindo os requisitos que ía cubrir o Sprint en cuestión (Sprint Planning), revisando as funcionalidades implementadas unha vez rematadas (Sprint Review) e identificando os aspectos a mellorar para o seguinte sprint (Sprint Retrospective).

Capítulo 4

Ánalise de requisitos global

No capítulo actual describiranse os roles que actúan dentro da aplicación así como os requisitos globais da mesma.

A captura de requisitos realizouse mediante historias de usuario, tal como se explicou no apartado de metodoloxía.

4.1 Roles

Os actores que interactúan co sistema diferéncianse en función das accións que poden levar a cabo dentro da aplicación, distinguindo desta maneira dous roles:

- **Usuario sen autenticar**: Son aqueles usuarios que non teñen conta no sistema ou simplemente non fixeron uso dela para autenticarse. Só teñen acceso as pantallas de rexistro e autenticación.
- **Usuario autenticado**: Dise dos usuarios que realizaron a autenticación con éxito aportando o seu nome de usuario e contrasinal.

  Este tipo de usuarios pode utilizar todas as funcionalidades do sistema agás a de autenticarse e rexistrarse.

4.2 Historias de usuario

Nesta sección móstrase a lista de historias de usuario que conforman o Product Backlog da aplicación. Na táboa 4.1 podemos observar o contido do Product Backlog.

As historias de usuario permiten concentrar nunha frase a funcionalidade que representan. Polo contrario, as veces non é suficiente con esta breve descrición para comprender a funcionalidade completa do requisito, polo que ademais da lista tamén se adxunta unha descrición máis detallada por cada unha delas.

ID	Nome	Descrición
RF-01	Autenticación	Proceso de autenticación no sistema por parte dun usuario
RF-02	Rexistro	Proceso de rexistro no sistema por parte dun usuario
RF-03	Búsqueda	Búsqueda de lugares en función dunha serie de filtros
RF-04	Mostrar lugares	Mostrar unha lista de lugares resultado dunha búsqueda previa
RF-05	Ver detalles dun lugar	Mostrar os detalles dun lugar concreto xunto coas puntuacións que os usuarios fixeron sobre el e as accións que se poden realizar
RF-06	Puntuar	Puntuar un lugar en función da accesibilidade que o usuario percibe
RF-07	Marcar como favorito	Marcar un lugar como favorito do usuario
RF-08	Ver lugares favoritos	Recuperar os lugares que o usuario marcou como favoritos
RF-09	Engadir lugar	Realizar unha petición contra o sistema para engadir un novo lugar
RF-10	Votar positivamente unha petición	Votar positivamente unha petición realizada para engadir un novo lugar ao sistema
RF-11	Votar negativamente unha petición	Votar negativamente unha petición realizada para engadir un novo lugar ao sistema
RF-12	Ver peticións de novos lugares	Ver a lista de peticións que os usuarios fixeron para engadir novos lugares ao sistema
RF-13	Ver detalles dunha petición	Ver os detalles dunha petición concreta xunto coas accións que pode realizar o usuario sobre ela
RF-14	Realizar petición de borrado	Realizar unha petición para que se borre un lugar concreto
RF-15	Ver peticións de borrado	Ver a lista de petitións de borrado de lugares que se atopan no sistema
RF-16	Realizar petición de modificado	Realizar unha petición para que se modifiquen certos campos dun lugar
RF-17	Ver peticións de modificado	Ver a lista de petitións de modificado de lugares que se atopan no sistema
RF-18	Modificar comentario	Modificar o texto dun comentario en concreto
RF-19	Borrar puntuación	Eliminar a puntuación e máis o comentario asociado (se o tivera)
RF-20	Ver puntuacións	Ver as puntuacións realizadas sobre un lugar concreto xunto cos seus comentarios asociados (se os tivera)

Táboa 4.1: ProductBacklog ou Táboa de requisitos

4.2.1 Funcionalidades

RF-01 Autenticación

Para iniciar sesión no sistema, o usuario deberá introducir o seu nome de usuario e o seu contrasinal. No caso de introducir unhas credenciais válidas, o usuario pasará a estar autenticado na aplicación, en caso contrario mostrarase unha mensaxe de erro.

A partir do momento en que se autentica con éxito, o usuario terá acceso ás funcionalidades dispoñibles para usuarios autenticados.

RF-02 Rexistro

Para rexistrarse no sistema, un usuario deberá proporcionar un nome de usuario (que debe ser único e non ter sido usado antes no sistema), un e-mail (coa mesma condición que o nome de usuario), un contrasinal, un nome e os apelidos. O sistema realizará unha serie de validacións sobre os datos introducidos, e, en caso de ser correctos creará unha nova conta de usuario e redirixirá ao usuario ao formulario de autenticación. En caso contrario amosará unha mensaxe indicando o erro acontecido.

Unha vez rexistrado con éxito, o usuario terá a posibilidade de autenticarse para ter acceso as funcionalidades dispoñiles para usuarios autenticados.

RF-03 Búsqueda

A hora de realizar unha búsqueda na aplicación, o usuario poderá seleccionar unha serie de filtros que lle permitirán acoutar o ancho da búsqueda que desexa. Entre estes filtros podemos atopar: búsqueda por palabras clave comparando co nome dos lugares, filtro por distancia (en quilómetros) dende a posición actual do usuario e/ou filtrar só os lugares que o usuario teña marcados como favoritos.

Unha vez seleccionados os filtros que se desexan aplicar e executada a búsqueda, o sistema executará o caso de uso RF-04 Mostrar lugares.

RF-04 Mostrar Lugares

A partir do caso de uso RF-03 Búsqueda, o sistema mostrará unha pantalla co resultado da búsqueda en cuestión indicando para cada elemento o nome do lugar e en caso de existir máis de dez elementos, situará no final da pantalla un botón que permite ver os seguintes elementos agrupados de dez en dez.

En caso de facer uso do botón que mostra os seguintes dez elementos, o sistema mostrará outro botón que permite volver aos dez elementos anteriores (mostrando a súa vez o botón dos dez elementos seguintes en caso de que os houbera).

A aplicación permite ao usuario pulsar en cada un dos elementos que corresponde a un

lugar, executando desta maneira o caso de uso RF-05 Ver detalles dun lugar.

RF-05 Ver detalles dun lugar

A partir da selección dun lugar en concreto no caso de uso RF-04 Mostrar lugares, o usuario poderá ver todos os datos concretos sobre o lugar que escolleu. Entre eles estarán: o nome da localización, as coordenadas (latitude e lonxitude), a descrición, a presencia de ascensor, a existencia de rampa de acceso, a existencia de baños adaptados, a media de puntuación que os usuarios lle outorgaron a accesibilidade deste lugar, o número de puntuacións realizadas e un listado de puntuacións cos seus respectivos comentarios (se os tivera).

Ademais desta información, a pantalla mostrará ao usuario certos elementos que permiten executar algúns casos de uso, a continuación definiranse estes elementos. En primeiro lugar unha estrela na parte superior dereita da pantalla. Esta estrela ademais de indicar se o lugar en concreto se atopa na lista de lugares favoritos do usuario (en función da cor que mostre) tamén permite executar o caso de uso RF-07 Marcar como favorito.

A continuación, unha icona con forma de papeleira que permite ao usuario emitir unha solicitude de borrado sobre o lugar actual. Por outra parte está o caixón de comentarios, unha zona situada ao final da pantalla que permite ao usuario seleccionar unha puntuación e escribir un comentario sobre o lugar actual, pudendo executar desta maneira o caso de uso RF-06 Puntuar.

Para finalizar, o usuario ten a posibilidade de pulsar sobre a puntuación que el mesmo fixera sobre ese lugar e desta maneira poder executar os casos de uso RF-22 Modificar comentario ou RF-23 Borrar puntuación.

RF-06 Puntuar

A partir do caso de uso RF-05 Ver detalles dun lugar, un usuario pode puntuar o lugar en cuestión seleccionando para isto unha puntuación e podendo engadir ademais un comentario adicional que complemente esta puntuación.

Unha vez realizada a puntuación, en caso de que o usuario volva executar o caso de uso, tanto a puntuación como o corpo do comentario serán reemplazados pola puntuación máis actual. Ademais disto, tamén poderá modificar o corpo do comentario asociado a esa puntuación.

Unha vez realizada esta puntuación, tanto a media das valoracións dos usuarios como o número de puntuacións veranse actualizadas de maneira inmediata.

RF-07 Marcar/Desmarcar como favorito

A partir do caso de uso RF-05 Ver detalles dun lugar, un usuario pode marcar ou desmarcar o lugar actual como favorito, pulsando para isto na estrela que se atopa na parte superior

dereita da pantalla.

Unha vez executado o caso de uso, a estrela cambiará de cor indicando desta maneira a pertenza ou non a lista de lugares favoritos do usuario (amarela se pertence e baleira se non).

RF-08 Ver lugares favoritos

A través da pantalla que mostra o perfil do usuario autenticado no sistema, ese usuario en cuestión pode obter todos aqueles lugares que foran marcados por el como favoritos.

De igual maneira que en casos de uso anteriores, esta pantalla amosará un listado dos lugares marcados como favoritos identificándoos polo nome do lugar. Ademais, en caso de ter máis de dez lugares nesta lista, a pantalla mostará na parte inferior un botón que permite mostrar o seguinte grupo de lugares. Se o usuario fai uso desta funcionalidade, a aplicación mostrará un botón que permite recuperar os dez lugares anteriores (mostrando a súa vez o botón dos dez elementos seguintes en caso de que os houbera).

RF-09 Engadir lugar

A partir do menú da aplicación é posible acceder a este caso de uso, que permite cubrir unha serie de campos correspondentes a información que contería un lugar (nome, descrición, coordenadas e a existencia de rampa, ascensor e/ou baños adaptados) podendo desta maneira enviar unha solicitude ao servidor para que se poida engadir como un novo lugar no sistema.

Con respecto as coordenadas do lugar a engadir, existe a posibilidade de cubrir este valor sen necesidade de escribilo manualmente. Por un lado é posible usar un buscador por palabras que utiliza o motor de búsqueda de Google para proporcionar ao usuario suxerencias sobre os lugares que xa existen en Google Maps. Por outra parte, pódese seleccionar a ubicación a través dun mapa interactivo que modifica estes valores no momento no que selecciones un punto do mesmo.

RF-10 Votar positivamente unha petición

A partir do caso de uso RF-13 Ver detalles dunha petición, o usuario pode votar positivamente a petición actual a través dun botón situado na parte inferior dereita da pantalla.

Unha vez realizada a votación o usuario non poderá volver votar sobre esa petición nin retractarse do seu voto.

RF-11 Votar negativamente unha petición

A partir do caso de uso RF-13 Ver detalles dunha petición, o usuario pode votar negativamente a petición actual a través dun botón situado na parte inferior esquerda da pantalla.

Unha vez realizada a votación o usuario non poderá volver votar sobre esa petición nin retractarse do seu voto.

RF-12 Ver peticións de novos lugares

A partir do menú da aplicación é posible acceder a este caso de uso, que permite ver mediante un listado aquelas peticións (actualmente activas) correspondentes a adición de novos lugares no sistema.

En caso de ter máis de dez peticións nesta lista, a pantalla amosará na parte inferior un botón que permite mostrar o seguinte grupo de peticións. Se o usuario fai uso desta funcionalidade, a aplicación mostrará un botón que permite recuperar as dez peticións anteriores (mostrando a súa vez o botón dos dez elementos seguintes en caso de que os houbera).

RF-13 Ver detalles dunha petición

A partir da selección dunha petición en concreto nos casos de uso RF-12 Ver peticións de novos lugares, RF-17 Ver peticións de borrado e RF-19 Ver peticións de modificado, o usuario poderá ver todos os datos concretos sobre a petición que escolleu. Entre eles estarán: o nome da localización, as coordenadas (latitude e lonxitude), a descrición, a presencia de ascensor, a existencia de rampa de acceso, a existencia de baños adaptados, o número de votos positivos e o número de votos negativos.

Ademais desta información, a pantalla mostrará dous botóns na parte inferior referentes a votación (un para voto positivo e outro para voto negativo). De pulsares no botón de voto positivo executaríase o caso de uso RF-10 Votar positivamente unha petición, e en caso de pulsar o botón de voto negativo executaríase o caso de uso RF-11 Votar negativamente unha petición.

Unha vez realizada a votación, tanto o botón de votar positivamente como o de votar negativamente desaparecerán da vista do usuario, negándolle a posibilidade de votar máis dunha vez. Ademais disto, o número de votos verase actualizado de maneira automática.

RF-14 Realizar petición de borrado

A partir do caso de uso RF-05 Ver detalles dun lugar, a aplicación proporciona unha icona con forma de papeleira que permite ao usuario pulsar sobre ela para realizar unha petición de borrado sobre o lugar actual. Unha vez pulsado, a aplicación despregará un modal que preguntará ao usuario se desexa realizar unha petición de borrado sobre ese lugar, en caso afirmativo realizarase a petición e pecharase o modal, en caso contrario só se pechará o modal e non se fará nada.

En caso de existir unha petición de borrado sobre ese mesmo lugar a aplicación mostrará un erro.

RF-15 Ver peticións de borrado

A partir do menú da aplicación é posible acceder ao listado de peticións activas que mane-

xa o sistema. Unha vez nesta pantalla e mediante o uso dun botón na parte superior, é posible recuperar as peticións de borrado, xa que por defecto recupéranse as peticións para engadir novos lugares.

En caso de ter máis de dez peticións nesta lista, a pantalla amosará na parte inferior un botón que permite mostrar o seguinte grupo de peticións. Se o usuario fai uso desta funcionalidade, a aplicación mostrará un botón que permite recuperar as dez peticións anteriores (mostrando a súa vez o botón dos dez elementos seguintes en caso de que os houbera).

RF-16 Realizar petición de modificado

A partir do caso de uso RF-05 Ver detalles dun lugar, a aplicación proporciona unha icona con forma de lapis que permite ao usuario pulsar sobre ela para realizar unha petición de modificado sobre o lugar actual. Unha vez pulsado o usuario é redirixido a unha páxina na cal se poden observar os campos que poden ser modificados do lugar en cuestión, é dicir: o nome do lugar, a descrición, se posúe ascensor, se posúe rampa de acceso e se posúe baños adaptados.

É necesario modificar algún dos valores para que a aplicación permita lanzar a petición de modificado, en caso contrario mostrarase un mensaxe de erro. Se o lugar sobre o que se intenta facer a petición xa ten unha petición de modificado aberta, mostrarase un erro acorde.

RF-17 Ver peticións de modificado

A partir do menú da aplicación é posible acceder ao listado de peticións activas que manexa o sistema. Unha vez nesta pantalla e mediante o uso dun botón na parte superior, é posible recuperar as peticións de modificado, xa que por defecto recupéranse as peticións para engadir novos lugares.

En caso de ter máis de dez peticións nesta lista, a pantalla amosará na parte inferior un botón que permite mostrar o seguinte grupo de peticións. Se o usuario fai uso desta funcionalidade, a aplicación mostrará un botón que permite recuperar as dez peticións anteriores (mostrando a súa vez o botón dos dez elementos seguintes en caso de que os houbera).

RF-18 Modificar comentario

A partir do caso de uso RF-05 Ver detalles dun lugar e situados no caixón de comentarios da parte inferior da pantalla, un usuario pode pulsar sobre a puntuación realizada por el (co seu correspondente comentario asociado se o tivera) despregando desta maneira un modal que entre outras cousas mostra unha icona con forma de lapis.

Se o usuario decide pulsar sobre a icona do lapis, a aplicación cambiará o modal por un caixón de texto que permitirá ao usuario modificar o texto do comentario (NON a puntuación) para despois decidir se gardalo ou descartar os cambios.

RF-19 Borrar puntuación

A partir do caso de uso RF-05 Ver detalles dun lugar e situados no caixón de comentarios da parte inferior da pantalla, un usuario pode pulsar sobre a puntuación realizada por el (co seu correspondente comentario asociado se o tivera) despregando desta maneira un modal que entre outras cousas mostra unha icona con forma de papeleira.

Se o usuario decide pulsar sobre a papeleira, a aplicación emitirá unha mensaxe de confirmación para aclarar se de verdade se desexa borrar esta puntuación (e con ela o seu comentario asociado). Ademais desta mensaxe aparecen dous botóns xusto debaixo, un que serve para confirmar o borrado da puntuación e outro para anular a acción.

Capítulo 5

Planificación

Neste capítulo explicarase como se realizou a planificación do proxecto mediante Sprints e para cada un deles a carga de traballo que se lle asignou. Ademais disto, tamén se adxunta ao final da sección un cálculo aproximado dos custos do proxecto.

5.1 Sprints

Nesta sección detállanse os Sprints definidos para a consecución do proxecto. Cabe indicar que debido a falta de coñecemento nas tecnoloxías e na posta en práctica da metodoloxía, nas primeiras iteracións non foi posible realizar unha planificación dos Sprints. Sen embargo, unha vez realizados os dous primeiros Sprints, xa se posuía a práctica e o coñecemento suficiente como para poder estimar e planificar.

5.1.1 Sprint 0: Investigación e formación

Debido ao descoñecemento sobre todas as tecnoloxías que se ían utilizar na consecución do proxecto, a finalidade deste Sprint foi a de formarse sobre estas tecnoloxías e máis a investigación dos diferentes compoñentes e recursos dispoñibles para o seu futuro uso.

5.1.2 Sprint 1: Posta en marcha

O obxectivo principal deste Sprint é a construción da estrutura do proxecto, é dicir, no caso de Spring Boot un arquetipo Maven que inclúa as dependencias necesarias para o noso sistema, e no de React Native unha estrutura de ficheiros adecuada. Así, a partir destas estruturas, iranse incorporando as futuras funcionalidades.

5.1.3 Sprint 2: Tratamento de usuarios

Nesta iteración realizaranse aquelas funcionalidades relacionadas co tratamento dos usuarios. É dicir, os requisitos:

- **RF-01: Autenticación**.
- **RF-02: Rexistro**.

Esta iteración serve como toma de contacto coas novas tecnoloxías, posto que ata o momento só se realizaron as estruturas de ambas pero non se interactuou con elas.

5.1.4 Sprint 3: Búsqueda e favorito

Para a consecución deste Sprint implementáronse aquelas funcionalidades que teñen que ver coa búsqueda de lugares, o visionado do resultado da mesma, a mostra dos detalles dun dos lugares concretos e a posibilidade de marcar un deses lugares como favorito para o usuario. É dicir, na táboa de requisitos correspondería con:

- **RF-03: Búsqueda**.
- **RF-04: Mostrar Lugares**.
- **RF-05: Ver detalles dun lugar**.
- **RF-07: Marcar como favorito**.

É necesario indicar que a implementación da historia RF-03: Búsqueda non se realizou de maneira completa nesta iteración, posto que ao ser unha funcionalidade tan completa e que depende de outras, nesta primeira aproximación unicamente se realizou unha funcionalidade básica que permite a búsqueda por palabras, engadindo filtros a medida que avanzan as iteracións.

5.1.5 Sprint 4: Comentarios

Neste Sprint leváronse a cabo aquelas tarefas relacionadas cos comentarios que unha puntuación leva asociados. Os requisitos implementados foron os seguintes:

- **RF-06: Puntuar (antes RF-06: Comentar**.
- **RF-18: Modificar comentario**.
- **RF-19: Borrar puntuación (antes RF-19: Borrar comentario**.

- **RF-20: Ver puntuacións (antes RF-20: Ver comentarios**.

Nun primeiro momento o requisito RF-06: Puntuar facía referencia soamente ao acto de comentar un lugar, pero máis tarde foi modificado incluíndo así os comentarios dentro da acción de puntuar. De igual maneira, o requisito RF-20: Ver puntuacións que nun primeiro lugar só mostraba os comentarios (só o texto, sen a puntuación) realizados sobre un lugar en concreto, logo foi modificado cando os comentarios foron unha asociación das puntuacións realizadas polos usuarios, mostrando desta maneira as puntuacións e os seus comentarios asociados.

5.1.6 Sprint 5: Puntuar e lugares favoritos

Nesta iteración realizáronse aquelas funcionalidades relacionadas coa puntuación dos lugares unha vez se modificou o requisito incluíndo desta maneira os comentarios como unha asociación das puntuacións. Ademais disto, tamén se implementou a funcionalidade que permite recuperar e ver os lugares favoritos dun usuario. Os requisitos asociados que se levaron a cabo foron os seguintes:

- **RF-08: Ver lugares favoritos**.
- **RF-06: Puntuar**.
- **RF-20: Ver puntuacións**.

5.1.7 Sprint 6: Peticións

Para a consecución deste Sprint realizáronse aquelas funcionalidades referentes ao tratamento das peticións no sistema, é dicir, as peticións para engadir lugares, modificalos e/ou eliminalos. Os requisitos asociados que se levaron a cabo foron os seguintes:

- **RF-09: Engadir lugar**.
- **RF-10: Votar positivamente unha petición**.
- **RF-11: Votar negativamente unha petición**.
- **RF-12: Ver peticións de novos lugares**.
- **RF-13: Ver detalles dunha petición**.
- **RF-14: Realizar petición de borrado**.
- **RF-15: Ver peticións de borrado**.

- **RF-16: Realizar peticións de modificado**.
- **RF-17: Ver peticións de modificado**.

5.1.8 Sprint 7: Elaboración da memoria e mellora da aplicación

O último Sprint deste proxecto foi adicado a elaboración desta memoria. Ademais desta tarefa, aproveitouse a realizar diferentes melloras e modificacións na aplicación tendo en conta os aspectos identificados na Sprint Review da iteración anterior.

5.2 Cálculo de custos

Para o cálculo de custos deste proxecto terase en conta as horas empreadas na consecución do mesmo, as cales se poden consultar na táboa 5.1. Como valor para o custo medio por hora tomarase 25 euros.

Sprint	Obxectivo	Data Inicio	Data Fin	Horas
0	Investigación e Formación	10/11/2018	08/01/2019	160
1	Posta en marcha	08/01/2019	22/01/2019	100
2	Tratamento dos usuarios	22/01/2019	04/02/2019	120
3	Búsqueda e favorito	04/02/2019	23/04/2019	50
4	Comentarios	23/04/2019	29/05/2019	70
5	Puntuar e lugares favoritos	29/05/2019	24/06/2019	80
6	Peticións	24/06/2019	24/08/2019	80
7	Elaboración da memoria e mellora da aplicación	24/07/2019	28/08/2019	150
TOTAL		10/11/2018	28/08/2019	810

Táboa 5.1: Resumo de horas do proxecto

Polo tanto, tendo en conta o total de horas empregadas e máis o valor medio do custo por hora, o custo aproximado do proxecto sería:

Custo total = 810 horas * 25 euros/hora = **25.250 €**

Capítulo 6

Fundamentos tecnolóxicos

A continuación abordaranse aquelas tecnoloxías que foron utilizadas durante o desenvolvemento do proxecto. Para a súa mellor comprensión, dividiranse estas tecnoloxías en tres grupos:

- **Backend:** Tecnoloxías utilizadas na implementación do backend.
- **Frontend:** Tecnoloxías utilizadas na implementación do frontend.
- **Outras:** Tecnoloxías utilizadas de maneira adicional durante o desenvolvemento do proxecto.

6.1 Backend

Para desenvolver o backend da aplicación decantouse por unha API REST implementada con Spring Boot. Seguidamente serán mencionadas e explicadas cada unha das tecnoloxías que foron necesarias para a implementación do backend.

6.1.1 REST

REST [4] é un conxunto de principios de arquitectura que se utiliza para describir calquera interface entre sistemas que utilice HTTP para tratar un conxunto de datos independentemente do formato no que se atope (XML, JSON, etc).

Un servizo REST expón unha serie de recursos e un conxunto de operacións sobre eles para poder manipulalos. Ao tratarse de sistemas independentes (o servizo REST e o sistema que o consome), existe unha separación entre o cliente e o servidor, o que evita dependencias entre ambos.

6.1.2 Java

Java [5] é unha linguaxe de programación de propósito xeral, concorrente e orientada a obxectos que foi deseñada para ter o mínimo número de dependencias de implementación posibles.

A intención da súa creación é que os programadores escriban o código unha vez e o executen en calquera dispositivo ("write once, run anywhere"), o que quere dicir que o código que é executado nunha plataforma non ten que ser recompilado para ser executado noutra, sempre e cando teña soporte para a máquina virtual de Java (JVM).

6.1.3 JUnit

JUnit [6] é un conxunto de bibliotecas creadas coa finalidade de realizar probas unitarias en aplicacións Java. O seu funcionamento é o seguinte: JUnit permite realizar a execución de clases Java de maneira controlada, avaliando se o funcionamento de cada un dos métodos da clase compórtase como se espera. É dicir, en función duns valores de entrada avalíase un valor de retorno esperado; En caso de que os métodos cumpran coa especificación, JUnit devolverá que o método da clase pasou exitosamente a proba; en caso de que o valor esperado sexa diferente ao que devolveu o método durante a execución, JUnit devolverá un erro referenciando o método correspondente.

6.1.4 Maven

Maven [7] é unha ferramenta de software para a xestión e construción de proxectos Java. O seu funcionamento baséase no POM, un modelo de configuración de construción que non é máis ca un ficheiro XML que se utiliza para describir o proxecto a construír, as súas dependencias con outros módulos e a orde de construción dos seus elementos.

Algo moi importante de Maven é que está preparado para utilizar a rede. Isto quere dicir que o motor incluído no seu núcleo pode descargar plugins dun repositorio de maneira dinámica.

Maven simplifica moitas tarefas típicas dun proxecto Java, como poden ser:

- Automatización do compilado.
- Automatización do empaquetado software.
- Centralización da xestión de dependencias.
- Posibilidade de definir múltiples perfís de configuración.

6.1.5 Spring Boot

Spring Boot [2] é unha ferramenta que facilita a creación de aplicacións independentes baseadas en Spring que directamente podes executar sen necesidade de despregar un servidor web, xa que a propia ferramenta prové dun propio. Ademais, Spring Boot conta cun complexo módulo de autoconfiguración que permite que o desenrolador simplemente execute a aplicación sen necesidade de definir previamente nada. Isto permite que o programador poida centrarse na tarefa de codificar deixando de lado outras tarefas referentes a infraestrutura.

6.1.6 PostgreSQL

PostgreSQL [8] é un sistema de xestión de base de datos relacional orientado a obxectos e de cógido aberto. Posúe diversas características que o fixeron idóneo a hora de escollelo como sistema de xestión de base de datos, como son:

- **Alta concurrencia:** Mediante un sistema denominado MVCC (acceso concorrente multiversión, polas súas siglas en inglés) PostgreSQL permite que mentres un proceso escribe nunha táboa, outros accedan a mesma táboa sen necesidade de bloqueos. Cada usuario obtén unha versión consistente dos datos.
- **Amplia variedade de tipos nativos:** Posúe gran variedade de tipos nativos, entre os que se atopan: texto de largo ilimitado, direccións ip, etc.
- **Funcións:** Trátanse de bloques de código que se executan na parte do servidor. Poden ser escritos en varias linguaxes, coa potencia que cada un deles da. Esta característica en concreto foi de moita axuda a hora de realizar as búsquedas por distancia no servizo Rest, xa que se implementou unha función que simplificaba esta tarefa.

6.2 Frontend

6.2.1 JavaScript

JavaScript (abreviado comunmente JS) [9] é unha linguaxe de programación interpretada, dialecto do estándar ECMAScript [10], debilmente tipada e dinámica.

O estándar de JavaScript empregado ao longo deste proxecto será o ECMAScript 2018. Esta versión de ECMAScript inclúe múltiples características que simplifican en gran medida o desenvolvemento en JavaScript. Algunhas das máis destacadas son as funcións frecha (arrow functions), o uso de clases (sintactic sugar, non unha clase como se coñece en Java), a inclusión de constantes e variables locais (en lugar do tradicional ”var”) ou a inclusión de construtores async/await.

6.2.2 React Native

React Native [1] é un framework "open-source" creado por Facebook que permite desenvolver aplicacións para Android, IOS e UWP mediante o uso de React xunto coas características propias da plataforma nativa. En esencia React Native é unha librería para JavaScript que ofrece certas facilidades como pode ser a sintaxe JSX.

Non é un Webview, todo o que se executa é absolutamente nativo. As nosas aplicacións non corren nun navegador, polo que non temos certos elementos que si atopamos en React como poden ser DOM, CSS, etc.

Esta tecnoloxía, en términos intelixibles, practicamente ofrece un tradutor que converte o código nativo escrito en React Native a Objective-C para poder ser executado en iOS e a Java para poder ser executado en Android. Polo tanto, a experiencia do usuario é idéntica a dunha aplicación nativa.

Os compoñentes

Os compoñentes, ao igual que en React [11], son as unidades a partir das cales se forman as aplicacións. Permiten estruturar a aplicación mediante pezas independentes e reusables, obtendo así partes totalmente funcionais que poden ser utilizadas de maneira independente. Conceptualmente, son funcións JavaScript que reciben unha entrada (props) e devolven un elemento React.

Os compoñentes a súa vez poden estar formados por outros compoñentes, o que nos permite crear compoñentes moi complexos a partir de outros moi sinxelos, proporcionando así unha árbore de composición.

En React (e por ende en React Native) podemos diferenciar dous tipos de compoñentes en función do seu comportamento interno: compoñentes funcionais e compoñentes de clase. Os compoñentes de clase ou con estado son aqueles que permiten manter datos propios ao largo do tempo e implementar comportamentos nos seus diferentes métodos do ciclo de vida. Polo contrario, un compoñente funcional carece de estado, polo que o seu comportamento é limitado e adoita utilizarse unicamente como contenedor.

O estado é mutable e privado para cada compoñente. Non debe manipularse directamente, senón mediante a función setState(), pois de non ser así podería provocar resultados inesperados.

```
1
2 //Compoñente funcional (Sin estado)
3 function Welcome(props) {
4   return <Text>Hello, {props.name}</Text>;
5 }
6
7 //Compoñente de clase (Con estado)
```

```
class Welcome extends React.Component {
  render() {
    return <Text>Hello, {this.props.name}</Text>;
  }
}
```

JSX

JSX é unha sintaxe de etiquetas que se pode definir como unha extensión da sintaxe de JavaScript. Pode parecer un simple linguaxe de marcado coma no caso de HTML, pero mantén todas as funcionalidades propias de JavaScript. A continuación adxúntase un fragmento de código no que podemos ver a diferencia de utilizar JSX e non utilizalo.

```

// Con JSX
const App = () => {
    return <div>Hello world!</div>
}

// Sen JSX
const App = () => {
  return React.createElement("div", null, "Hello world!");
};
```

Ciclo de vida en React Native

En React Native o ciclo de vida divídese en tres fases ben diferenciadas: Fase de construción (mounting), fase de actualización (updating) e fase de destrución (unmounting).

- **Fase de Construción (mounting):** Esta etapa lévase a cabo a primeira vez que o compoñente se vai montar, preparando as súas props, renderizándoo e montándoo.
- **Fase de Actualización (updating):** Esta fase dase en calquera momento que o compoñente sufra algunha modificación.
- **Fase de Destrución (unmounting):** Esta fase é executada cando un elemento é eliminado.

Nas figuras 6.1 e 6.2 podemos observar os métodos involucrados nas fases de construción e actualización do ciclo de vida. Non se adxunta unha figura que corresponda a fase de destrución posto que está so inclúe o método componentWillUnmount().

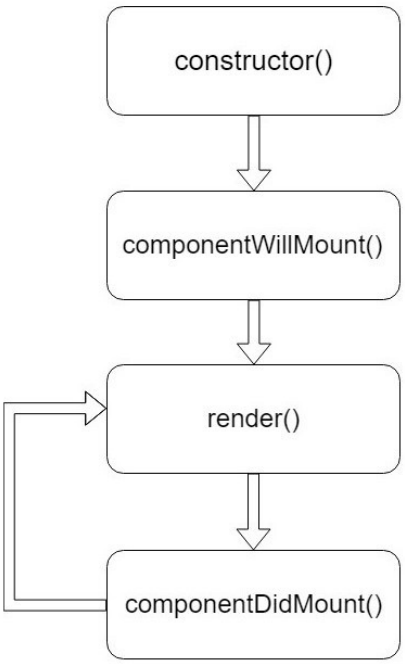

Figura 6.1: Fase de construcción(mounting)

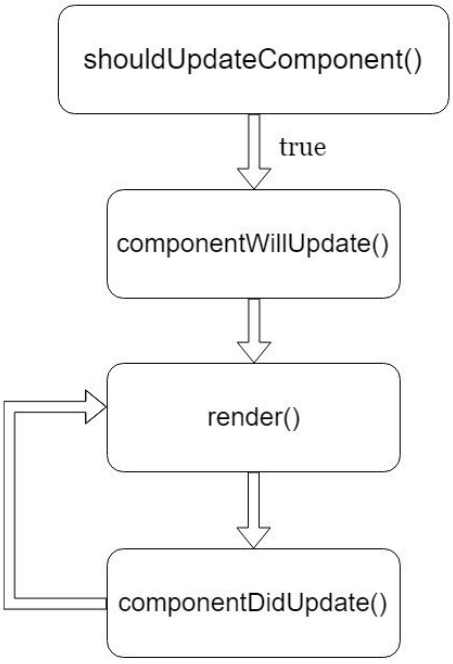

Figura 6.2: Fase de actualización(updating)

6.2.3 Redux

Redux [12] é un patrón de arquitectura de datos que permite manexar o estado da aplicación dunha maneira predecible. Axuda ao programador a escribir aplicacións que se comportan de maneira consistente, corren en distintos ambientes (cliente, servidor e nativo), e son fáciles de probar. Está pensado para reducir o número de relacións entre compoñentes da aplicación e manter un fluxo de datos sinxelo.

En resumo, estes serían os beneficios que aporta a aplicación do patrón de Redux:

- Arquitectura escalable de datos.
- Maior control sobre o fluxo de datos e o estado da aplicación.
- Estado global e inmutable.

A continuación describiranse brevemente os compoñentes que conforman Redux:

Estado

En Redux, todo o estado da aplicación almacénase como un único obxecto polo que é unha boa idea pensar na súa forma antes de comezar a programar. Os creadores de Redux aconsellan deseñalo de forma semellante a unha base de datos, e mantelo tan normalizado como sexa posible.

O estado en Redux é inmutable, polo que non podemos modificalo directamente, só podemos ler del para representalo na vista e en caso de querer modificalo debe facerse a través de reducers e accións.

Accións

Unha acción é, simplemente, un obxecto JavaScript que inclúe polo menos un atributo "type" que indica o tipo de acción que estamos emitindo e en caso de que existan datos asociados ao cambio ou modificación, un atributo payload con eses datos.

```

//Exemplo de acción en Redux
{
  type: ADD_TODO,
  text: 'Build my first Redux app'
}

//Código extraido da páxina web de Redux
```

Reducers

Os Reducers especifican como cambia o estado da aplicación en resposta ás accións enviadas ao Store. É necesario recordar que as accións só describen o que sucede, pero non describen como cambia o estado da aplicación. Os reducers, son funcións puras (non realizan efectos colaterais) que toman o estado anterior e máis unha acción e devolven un novo estado.

As aplicacións teñen un único estado, pero isto non implica que deba existir un único Reducer. O máis aconsellable é ter un conxunto de reducers e que cada un se ocupe de manter unha parte do estado. Sen embargo, para poder facer uso de varios reducers, é necesario crear un reducer que combine ao resto, para isto utilízase a utilidade proporcionada por Redux chamada combineReducers() que o que fae é xerar unha función que chama a todos os reducers coa parte do estado seleccionada de acordo a súa propiedade e combinar os seus resultados nun único obxecto.

```

//Exemplo dun reducer manexando unha acción
function todoApp(state = initialState, action) {
  switch (action.type) {
    case SET_VISIBILITY_FILTER:
      return Object.assign({}, state, {
        visibilityFilter: action.filter
      })
    default:
      return state
  }
}

//Código extraído da páxina web de Redux
```

Store

Nas seccións anteriores definíuse o que eran as accións e os reducers, sendo o Store o obxecto que os reúne a ambos. O Store ten as seguintes responsabilidades:

- Contén o estado da aplicación.
- Permite o acceso ao estado a través de getState().
- Permite que o estado sexa actualizado a través de dispatch(action).
- Rexistra os listeners a través de subscribe(listener).

É importante destacar que nunha aplicación Redux só existirá un Store, pois cando se desexa dividir a lóxica para o manexo de datos, utilizarase a composición de reducers en lugar de moitos Stores.

Fluxo de datos

A arquitectura de Redux xira en torno a un fluxo de datos estritamente unidireccional. O ciclo de vida dos datos en calquera aplicación Redux sigue os seguintes catro pasos:

- Realízase unha chamada a dispatch(action).
- O store invoca ao reducer indicado, que recibe a acción xunto co estado da aplicación e calcula o novo estado.
- O reducer principal pode combinar a saída de todos os reducers nun único estado.
- O store garda o estado devolto polo reducer principal. Neste punto serian invocados os listeners que foran asociados mediante store.subscribe(listener).

6.2.4 Integración de Redux e React Native

A hora de utilizar Redux xunto coa nosa aplicación, debemos buscar a maneira de trasladar a parte necesaria da Store ás props. Para isto, simplemente debemos utilizar a función connect() de react-redux a hora de crear o compoñente.

connect([mapStateToProps], [mapDispatchToProps], [mergeProps], [options]) -> ContainerComponent

Todos os argumentos da función son opcionais, e os máis usados son as funcións mapStateToProps e mapDispatchToProps, as cales se explicarán a continuación.

- **mapStateToProps(state, [ownProps]): stateProps:** Se este argumento está presente, o compoñente subscribirase ás modificacións que sufra o estado a través do Store, e a función mapStateToProps executarase cada vez que este se actualice. O resultado de dita execución é un conxunto de props que se incorporarán ás do compoñente.
- **mapDispatchToProps(dispatch, [ownProps]):dispatchProps:** Recibe como argumento o método dispatch do store e devolve a lista de callbacks que se desexen inxectar no compoñente en cuestión (fusionándose coas props). Desta forma proporciona un mecanismo para disparar accións de Redux desde o compoñente, permitindo responder así a eventos.

Na figura 6.3 podemos observar a arquitectura de Redux tal e como a acabamos de explicar e como se relaciona cos compoñentes de React Native (no diagrama correspóndese coa parte de View).

6.2.5 npm

Npm [13] é un xestor de paquetes para NodeJS [14], un entorno de execución para JavaScript. Permite xestionar as dependencias entre as distintas librerías dun proxecto.

Cando é utilizado unicamente como xestor de dependencias, Npm permite instalar todas as dependencias dun proxecto cun simple comando. Todas as dependencias que se instalan son as que aparecen indicadas no arquivo "package.json", onde cada unha delas pode especificar unha versión concreta ou un rango válido.

6.3 Outras

6.3.1 Git

Git [15] é un software de control de versións pensado para manter dunha maneira eficiente e confiable as versións de aplicacións cando estas teñen un gran número de arquivos de código fonte. O seu propósito é o de levar rexistro dos cambios en arquivos compartidos e coordinar o traballo dun equipo que pode realizar modificacións sobre eles.

Unha das características máis importantes que identifica a Git é que se trata dun sistema distribuído no que cada usuario posúe unha copia en local do histórico de versións do proxecto. Os cambios remotos impórtanse como ramas adicionais que poden ser fusionadas de igual maneira que unha rama local.

6.3.2 GitHub

GitHub [16] é unha empresa que proporciona aloxamento para proxectos de desenvolvemento software utilizando o sistema de control de versións Git.

Este servizo ofrece unha gran variedade de ferramentas e funcionalidades como poden ser:

- Soporte para repositorios públicos e privados de forma gratuita.
- Wiki colaborativa para proxectos.
- Sistema de xestión de incidencias.
- Visor de ramas de traballo.
- Gráficos sobre diferentes parámetros (commits, colaboradores...).
- Pull Requests con Code Review e comentarios.

Ademais, GitHub ofrece integración con unha amplia variedade de servizos como poden ser ferramentas de inspección continua (P.E: SonarLint [17]) ou integración continua (P.E: Jenkins [18]).

6.3.3 Redmine

Redmine [19] é unha ferramenta para a xestión de proxectos que permite aos usuarios de diferentes proxectos realizar o seguimento e a organización dos mesmos. É posible optimizar o seu funcionamento agregando funcionalidades de diversos tipos.

Como sistema de xestión de proxectos inclúe unha gran variedade de ferramentas para facilitar as tarefas ao usuario. Entre elas podemos atopar:

- Sistema de seguimento de incidencias e erros.
- Calendario de actividades.
- Diagrama de Gantt.
- Wiki para os proxectos.
- Foro para colaboradores.
- Visor do repositorio do SCM asociado (se o tivera).

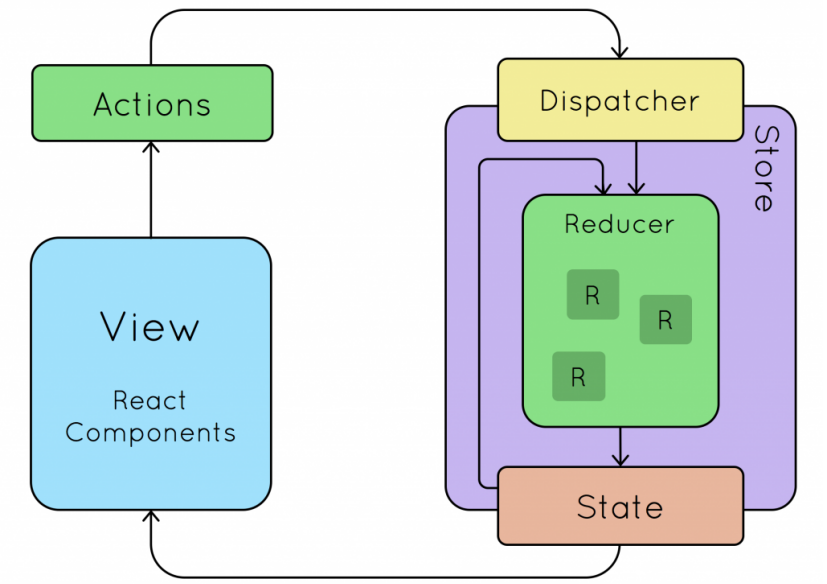

Figura 6.3: Integración de Redux con React Native

Capítulo 7

Desenvolvemento iterativo

No capítulo actual procederase a explicar os aspectos máis relevantes do desenvolvemento do proxecto, entrando en detalles de deseño e implementación para cada unha das iteracións realizadas.

7.1 Modelo de datos

Na figura 7.1 podemos observar o modelo de datos da nosa aplicación. Como se pode apreciar o número de entidades que trata non é moi elevado, máis a complexidade da aplicación radica na cantidade de operacións que se poden realizar sobre estas.

Con respecto as entidades que funcionan como peticións na aplicación (Petition, DeletePetition e ModifyPetition) decidiron modelarse de maneira separada debido a que a natureza de cada unha delas é moi diferente das demais, axudando así a que as operacións sobre elas sexan máis sinxelas.

Entre unha das clases do modelo de datos atópase unha chamada AuditModel, da cal herdan o resto das outras entidades. Esta clase posúe unha tarefa especial, xa que proporciona a toda aquela entidade que herde dela un seguimento sobre cando se creou e cando foi a última vez que se actualizou (engadindo dúas columnas a cada entidade na base de datos), aportando desta maneira trazabilidade sobre todas as entidades. No diagrama de clases optouse por duplicala para gañar así en claridade, pero a niveis prácticos unicamente existe unha clase con este nome.

7.2 Sprint 0: Investigación e Formación

Como comezo do proxecto e como xa se comentara antes, a primeira tarefa en levar a cabo foi unha formación sobre aquelas tecnoloxías escollidas para a consecución do mesmo.

Situándonos primeiro nas tecnoloxías referentes ao Backend podemos dicir que a curva

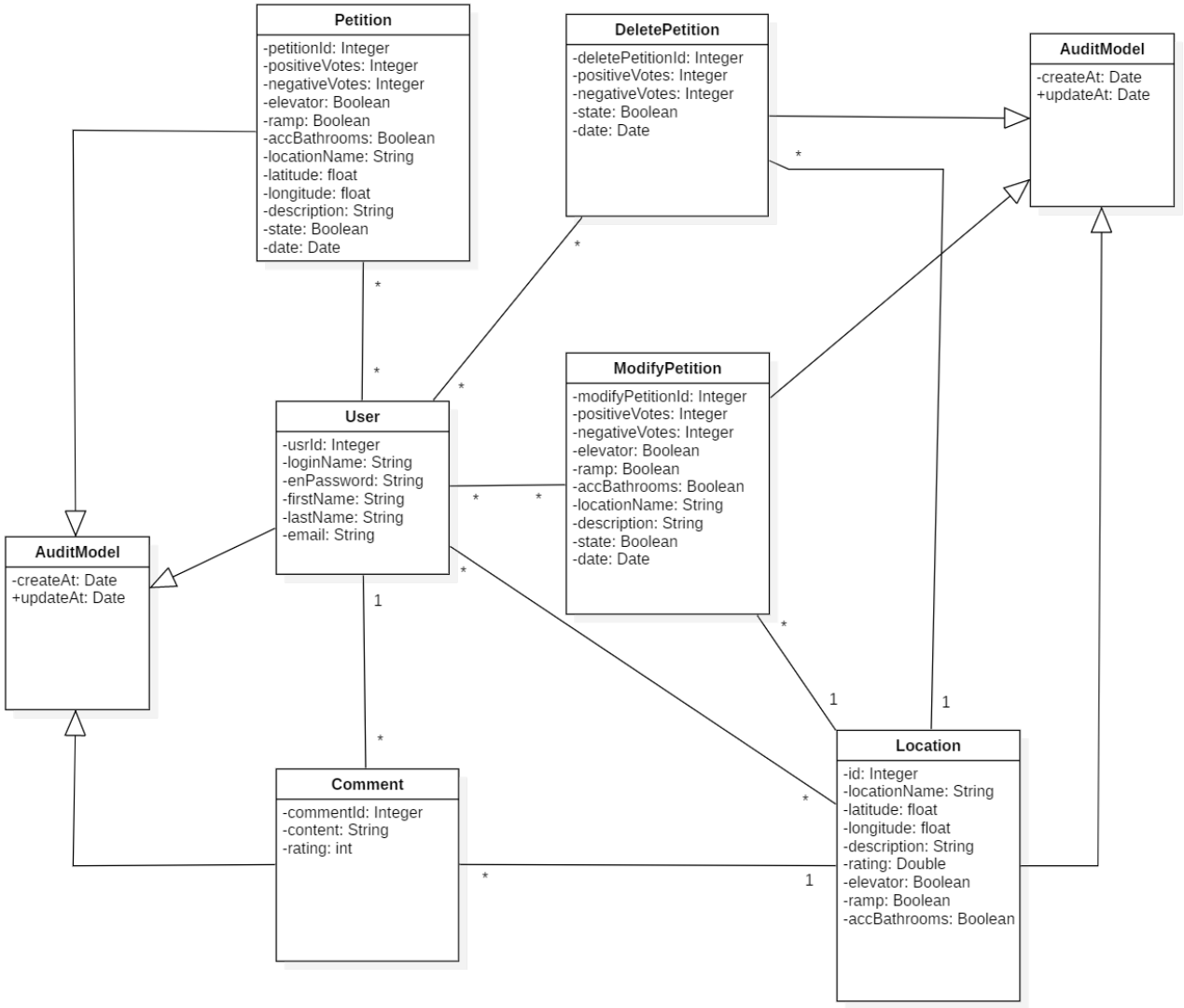

Figura 7.1: Modelo de datos do servizo

de aprendizaxe foi menor debido a que durante o transcurso do grado realizáronse varias implementacións de servizos Rest. A pesar disto, non se coñecía Spring Boot [2], polo que houbo que aprender o seu funcionamento e informarse sobre as súas características. Por outra parte, PostgreSQL [8] xa fora utilizado en ocasións pasadas, polo que o seu uso (quitando certas funcionalidades descoñecidas das que houbo que formarse, como as funcións) foi arbitrario.

Sen embargo, con respecto as tecnoloxías utilizadas para o Frontend a situación é totalmente contraria. En primeiro lugar, os coñecementos en JavaScript eran totalmente nulos, polo que para comezar a formarse fíxose uso da páxina oficial de JavaSript [9] e máis adiante, cando o nivel xa era o básico, fíxose uso da páxina JavaScript.info [20], a cal posúe un titorial para aprender JavaScript dende o nivel máis básico ata un nivel avanzado.

Continuando coas tecnoloxías a estudar aparece React Native. Con respecto a esta tecnoloxía non se sabía absolutamente nada no momento de comezar, polo que para poder dar os primeiros pasos e saber de que se trataba fíxose uso da páxina oficial de React Native [1], a cal proporciona toda a documentación sobre o framework e ademais un pequeno titorial para facer o teu primeiro código. A hora de aprender sobre esta librería foi de gran axuda a formación recibida sobre React nas prácticas en empresa, xa que aínda que os compoñentes sexan diferentes, a filosofía é a mesma en ambas tecnoloxías.

Para finalizar coas tecnoloxías do Frontend cabe mencionar Redux. Para poder adquirir coñecemento sobre esta librería fíxose uso da súa propia páxina web [12], xa que nela podemos atopar gran cantidade de exemplos ademais de documentación.

7.3 Sprint 1: Posta en marcha

Neste Sprint levouse a cabo a construción da estrutura do proxecto, implementando para isto unha pequena funcionalidade de proba que permitiu iniciarse en todas as tecnoloxías mencionadas anteriormente. A estrutura do proxecto pode diferenciarse en dúas partes, a estrutura do servizo Rest e a estrutura da aplicación React Native.

7.3.1 Estrutura do servizo Rest

Para a creación da estrutura do servizo Rest fíxose uso da ferramenta Spring Boot CLI. Esta ferramenta proporciona a posibilidade de crear a estrutura dun proxecto Spring Boot, indicándolle para isto as dependencias que se desexan incluir, como pode ser o xestor de base de datos a utilizar.

Ao fin e ao cabo unha aplicación Spring Boot non deixa de ser un proxecto Maven coa súa estrutura típica, contendo esta os seguintes elementos:

- **src/main/java**: Este paquete contén o código fonte da aplicación. A súa estrutura de paquetes podémola observar na figura 7.2 onde podemos apreciar diferentes paquetes

como o paquete ”services” (servizos para cada unha das entidades definidas no servizo), o paquete ”model” (definición das entidades), o paquete ”jwt” (tratamento do jason web token), etc.

- **src/main/resources**: Inclúe diversos ficheiros de configuración, como poden ser os ficheiros application.properties ou application.yaml. Ademais, tamén atopamos un ficheiro de recursos chamado messages.properties, onde se almacenan as mensaxes a utilizar na aplicación co seu correspondente contido.
- **src/test/java**: Inclúe as clases que definen as probas da nosa aplicación.
- **src/test/resources**: Contén un arquivo application.properties coa configuración específica para executar os test (P.E: Base de datos a utilizar).

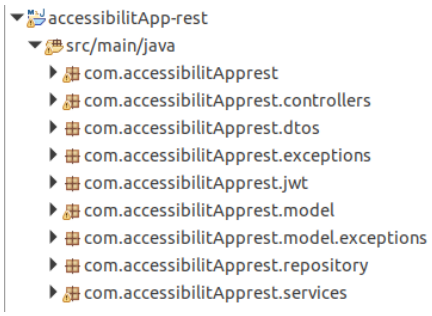

Figura 7.2: Estrutura de paquetes de /src/main/java

7.3.2 Estrutura da aplicación React Native

A hora de realizar a estrutura da aplicación móbil non se utilizou ningunha ferramenta que facilitase a súa creación como pode ser o caso de Expo [21], senón que se decidiu construír dende cero para así asegurar a máxima simplicidade na estrutura de ficheiros. Na figura 7.3 podemos observar a estrutura de ficheiros resultante.

A continuación procederase a explicar os ficheiros e directorios máis importantes para conseguir así unha mellor comprensión da estrutura do proxecto:

- **/android e /ios**: Directorios xerados automaticamente ao iniciar a aplicación, neles atópase o código nativo transpilado a partir do código escrito en JavaScript.
- **src/actions**: Neste directorio atópanse os ficheiros que fan referencia as accións e aos types de Redux como se explicou na sección 6.2.3.
- **src/backend**: Neste directorio podemos atopar todos aqueles ficheiros que fan referencia ao acceso a servizos do Backend. Ademais, para simplificar este acceso e facelo

máis intuitivo, creouse un ficheiro chamado ”appFetch” que proporciona a funcionalidade dun fetch de JavaScript [22] pero tendo en conta algúns aspectos como o tratamento de erros que no método básico non se teñen.

- **src/components e src/pages**: Aquí albérganse aqueles compoñentes que se utilizaron para a construción da aplicación. No directorio ”/components” decidiuse meter aqueles compoñentes de menos tamaño, deixando os compoñentes grandes para o directorio ”pages”, que contén compoñentes de maior tamaño e que só se utilizan nunha parte concreta da aplicación.

- **src/images**: Aquí almacénanse todas as imaxes das que se fai uso no sistema, xa sexan imaxes decorativas ou iconas dalgunhas funcionalidades.

- **src/reducers**: Neste directorio atópanse todos os reducers que utiliza Redux para xestionar o estado da aplicación. De igual maneira, isto foi explicado no apartado 6.2.3.

- **Routes.js**: A funcionalidade deste arquivo é a de enrutar todas as páxinas e compoñentes da nosa aplicación. Para isto, faise uso dunha librería de React Native chamada react-native-router-flux que utiliza por detrás outra tamén moi coñecida chamada react-navigation.

- **store.js**: Este ficheiro define o Store de Redux. Nel, atópase o reducer principal chamado ”rootReducer”, que centraliza a todos os demais definidos no directorio ”src/-reducers”. Este concepto de Store atópase definido no capítulo de tecnoloxías, máis concretamente na sección de Redux 6.2.3.

- **App.js**: Contén o compoñente de enrutamento da aplicación. Este compoñente é utilizado dende o arquivo ”index.js” para iniciar a aplicación.

- **app.json**: Ficheiro de configuración que define o nome da aplicación e cal é o nome que se vai mostrar.

- **index.js**: Pódese dicir que este arquivo constitúe o núcleo da aplicación, pois nel ínciase tanto o enrutamento da aplicación como Redux. Neste compoñente conéctanse React e Redux a través do compoñente <Provider/> e é aquí onde se inicializa o Store anteriormente mencionado. Ademais, faise unha chamada ao método ”init” do ficheiro ”appFetch”, o cal define o callBack de erro que se deberá lanzar en caso de que algo suceda mal (neste caso simplemente redirixir a unha páxina de erro). Na figura 7.4 podemos ver a forma que ten o ficheiro en cuestión.

- **package-lock.json e package.json**: Estes ficheiros definen todas as dependencias da aplicación que se descargarán ao facer uso de Npm 6.2.5.

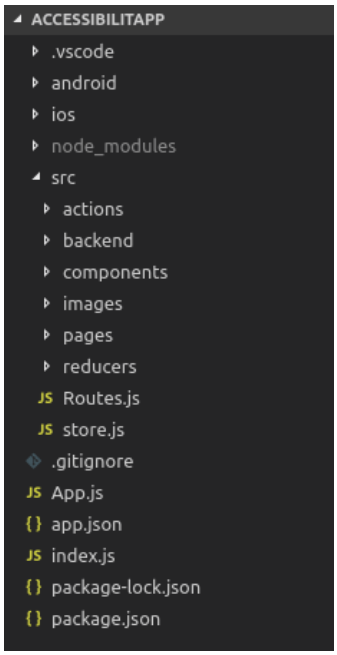

Figura 7.3: Estructura de ficheiros na aplicación móbil

```
const store = configureStore();

/* Configure backend proxy. */
backend.init(error => Actions.error());

const RNRedux = () => (
  <Provider store={store}>
    <App />
  </Provider>
);

AppRegistry.registerComponent(appName, () => RNRedux);
```

Figura 7.4: Código do ficheiro ”index.js”

7.4 Sprint 2: Tratamento de usuarios

Neste Sprint realizanse as funcionalidades relacionadas co tratamento do usuario. Para isto, escóllense as entradas correspondentes do Product Backlog descompoñéndoas en tarefas para conseguir así o Sprint Backlog.

7.4.1 Análise

Nesta sección defínese o aspecto do Sprint Backlog unha vez escollidas as entradas e definidas as tarefas. Para este Sprint escolléronse as historias de usuario RF-01: Autenticación e RF-02: Rexistro que se poden ver descompostas nas figuras 7.5 e 7.6 respectivamente.

RF-01 : Autenticación
Proceso de autenticación no sistema por parte dun usuario
Tarefas
Deseño do modelo de datos primixenio
Implementación da funcionalidade de autenticación no servizo Rest
Creación da vista de autenticación na aplicación
Integración da autenticación con Redux

Figura 7.5: Sprint 2: Autenticación

RF-02 : Rexistro
Proceso de rexistro no sistema por parte dun usuario
Tarefas
Implementación da funcionalidade de rexistro no servizo Rest
Creación da vista de rexistro na aplicación
Integración do rexistro con Redux

Figura 7.6: Sprint 2: Rexistro

7.4.2 Deseño

Nesta primeira iteración e partindo de que o sistema posuía unicamente a estrutura básica sen ningunha funcionalidade, foi necesario deseñar o modelo de datos inicial que unicamente inclúe a entidade ”User”. Na figura 7.7 podemos ver o aspecto inicial que posúe a entidade ”User”.

Para as tarefas deste Sprint foi necesario engadir o repositorio que accede aos datos da táboa ”User” na base de datos, ademais do seu Servizo e Controlador.

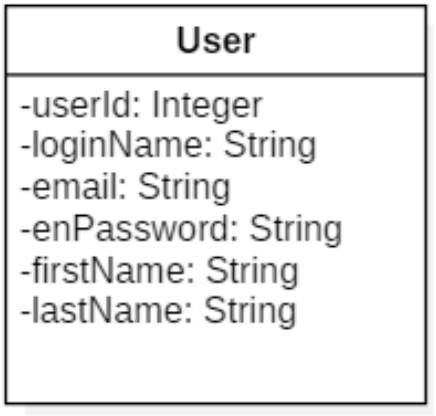

Figura 7.7: Entidade ”User” no modelo de datos inicial

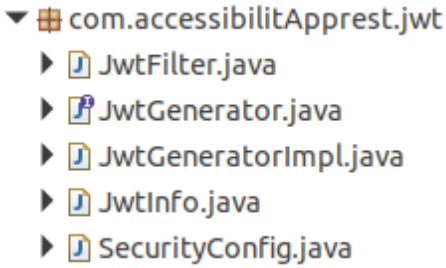

Figura 7.8: Clases incluídas no paquete JWT

Para a posterior implementación e uso de JWT na autenticación e rexistro, foi necesario engadir unha serie de clases que se engloban no paquete ”jwt” do servizo Rest. Na figura 7.8 podemos ver as clases mencionadas.

UserController é o controlador que recibe as peticións relacionadas co tratamento dos usuarios, este delega a lóxica de negocio na clase UserService que a súa vez delega as peticións a base de datos na clase UserRepository. Por outra parte, a hora de realizar a acción de autenticación, e unha vez comprobado que as credenciais son as correctas, xérase un JWT facendo uso da clase JwtGenerator, que xera un JWT a partir dun obxecto que contén a información necesaria para isto (este obxecto é unha instancia da clase JwtInfo).

Por outra parte, JwtFilter é unha clase que actúa como un filtro que intercepta as peticións recibidas polo servizo e realiza as comprobacións necesarias sobre o JWT. Por último, SecurityConfig é unha clase que contén configuración de Spring relativa a seguridade como poden ser as peticións as que se lle debe prohibir o acceso ou as restricións en función do usuario que as realice.

Na figura 7.9 podemos ver un diagrama onde se representan as clases anteriormente explicadas e a relación que gardan entre si.

RF-01 : Autenticación

Para a funcionalidade de Autenticación realizouse un deseño típico deste tipo de pantallas para que desta maneira sexa intuitivo para o usuario. Na figura 7.10 podemos observar o aspecto desta pantalla, mostrando os campos que o usuario debe cubrir coas súas credenciais, o botón para lanzar a acción de autenticarse e un enlace ao formulario de rexistro. O compoñente de React Native definido para isto é o chamado ”Login” do directorio ”src/pages”.

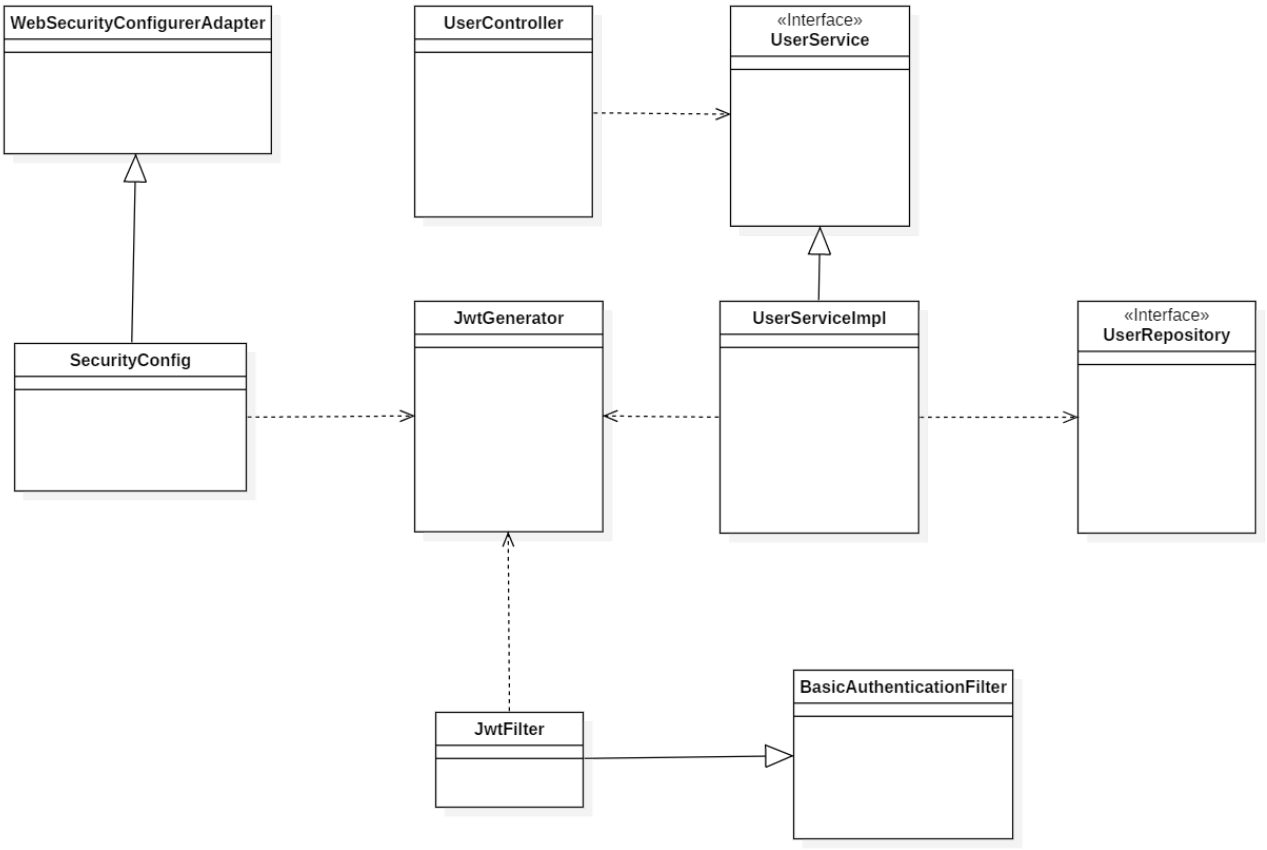

Figura 7.9: Diagrama de clases do paquete ”jwt”

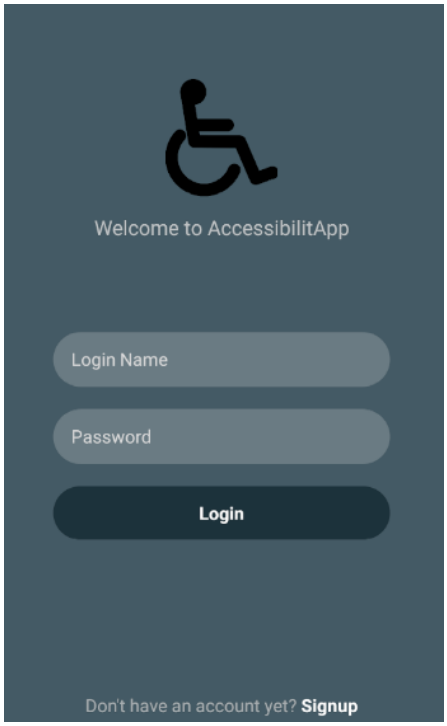

Figura 7.10: Vista do formulario de autenticación

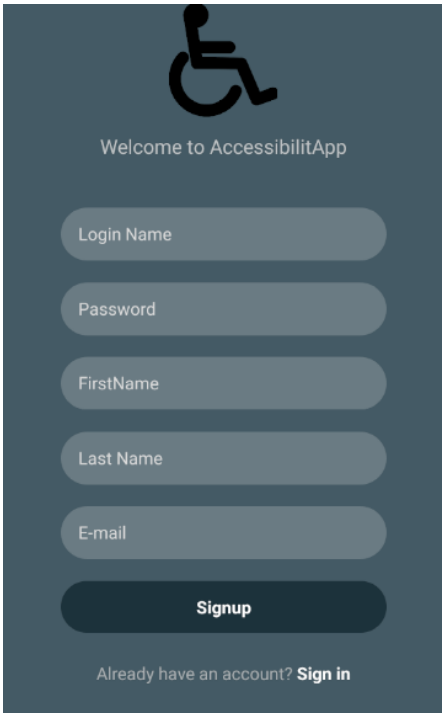

Figura 7.11: Vista do formulario de rexistro

RF-02 : Rexistro

Para a funcionalidade de Rexistro realizouse un deseño típico dunha pantalla de rexistro para que desta maneira sexa intuitivo para o usuario. Na figura 7.11 podemos observar o aspecto desta pantalla, mostrando os campos que o usuario debe cubrir coa información requirida, o botón para lanzar a acción de rexistrarse e un enlace ao formulario de autenticación. O compoñente de React Native definido para isto é o chamado ”Signup” do directorio ”src/pages”.

7.4.3 Implementación

Para a implementación da Autenticación e o Rexistro fíxose uso de JWT [23]. JSON Web Token é un estándar aberto baseado en JSON proposto para a creación de tokens de acceso que permiten a propagación de identidade e privilexios.

Grazas a JWT conseguimos que a noso servizo Rest non necesite manter un rexistro dos tokens de acceso, senón que é o usuario o que se encarga de almacenalo unha vez autenticado e envialo en cada unha das peticións que realice.

Na figura 7.12 podemos ver un diagrama de secuencias no que podemos comprobar como funciona o proceso de autenticación a través deste sistema.

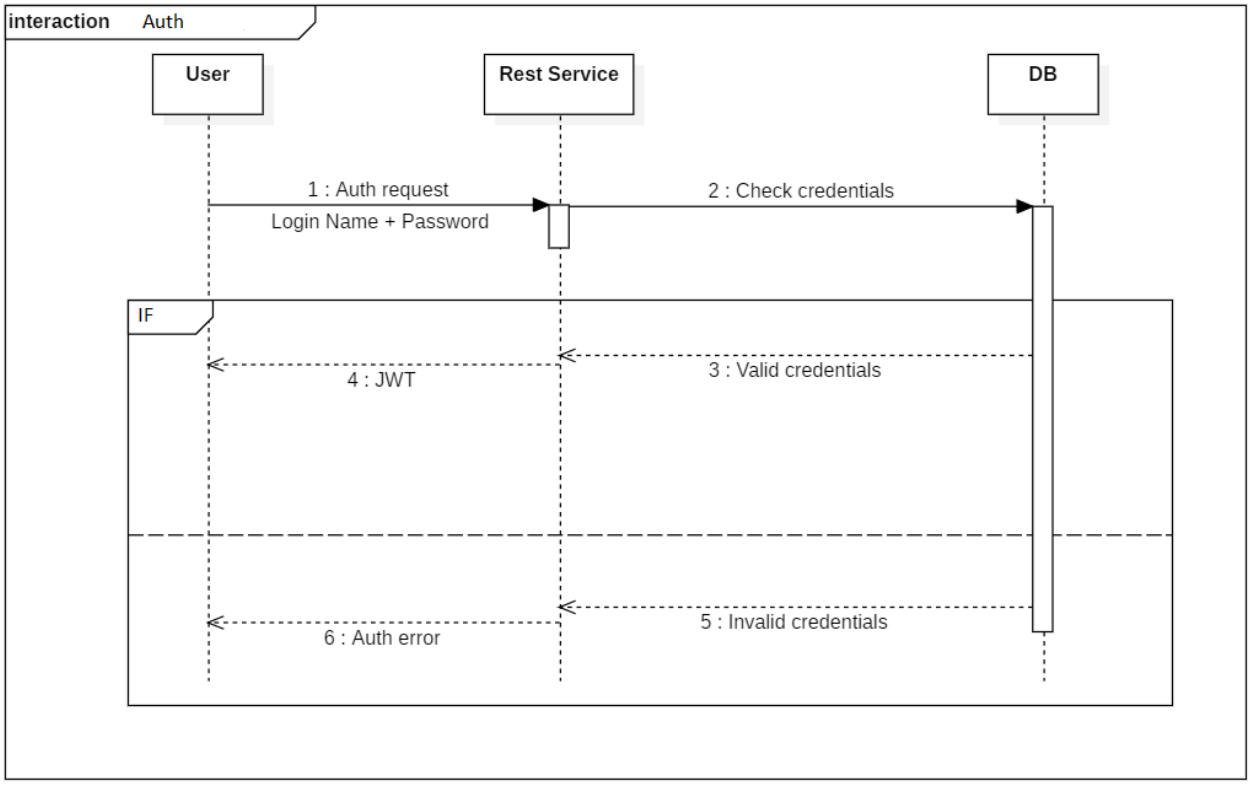

Figura 7.12: Diagrama de secuencias para a autenticación

7.5 Sprint 3: Búsqueda e favorito

Para a consecución deste Sprint impleméntanse aquelas funcionalidades que teñen que ver coa búsqueda de lugares, o visionado do resultado da mesma, a mostra dos detalles dun dos lugares concretos e a posibilidade de marcar un deses lugares como favorito para o usuario. Para isto, escóllense as entradas correspondentes do Product Backlog descompoñéndoas en tarefas para conseguir así o Sprint Backlog.

7.5.1 Análise

Nesta sección definirase o aspecto do Sprint Backlog unha vez escollidas as entradas e definidas as tarefas. Para este Sprint escolléronse as historias de usuario RF-03: Búsqueda, RF-04: Mostrar Lugares, RF-05: Ver detalles dun lugar e RF-07: Marcar como favorito que se poden ver descompostas nas figuras 7.13, 7.14, 7.15 e 7.16 respectivamente.

RF-03: Búsqueda
Búsqueda de lugares en función dunha serie de filtros
Tarefas
Modificar o deseño do modelo de datos
Implementación da funcionalidade de búsqueda no servizo Rest
Deseñar e crear a vista para a búsqueda de lugares
Engadir no estado de Redux o resultado da búsqueda co que iso implica (accións, types...)

Figura 7.13: Sprint 3: Búsqueda

RF-04: Mostrar Lugares
Mostrar unha lista de lugares resultado dunha búsqueda previa
Tarefas
Deseñar e crear a vista para mostrar a lista de lugares resultado da búsqueda

Figura 7.14: Sprint 3: Mostrar lugares

7.5.2 Deseño

Tendo en conta que na anterior iteración o modelo de datos só contiña a entidade ”User”, foi necesario ampliar este para así poder dar soporte a todas as funcionalidades relacionadas coas localizacións. Para isto, engadiuse unha nova entidade ”Location” que se relaciona coa

entidade ”User” polo feito de marcar unha localización como favorita por un usuario. Na figura 7.17 podemos ver o resultado do modelo de datos neste Sprint.

Para as tarefas deste Sprint foi necesario engadir o repositorio que accede aos datos da táboa ”Location” na base de datos, ademais do seu Servizo e Controlador.

RF-03 : Búsqueda

Para a funcionalidade de búsqueda fíxose un deseño que contén unha entrada de texto na parte superior complementada cos filtros correspondentes. Cabe destacar como xa se dixo anteriormente, que a medida que a aplicación foi evolucionando e foron aparecendo novos parámetros fóronse engadindo filtros a funcionalidade de búsqueda. Na figura 7.18 podemos observar o aspecto desta pantalla na súa primeira versión, mostrando a barra de texto na parte superior e o botón para lanzar a acción de buscar. O compoñente de React Native definido para isto é o chamado ”SearchPlaces” do directorio ”src/pages”.

RF-04 : Mostrar lugares

Para mostrar o resultado da búsqueda de lugares realizouse o deseño dunha pantalla que simplemente consiste nunha lista de elementos nos que se pode ler o nome da localización para cada un deles. En caso de ter máis de dez elementos aparecería na parte inferior da pantalla un botón que permite mostrar os seguintes dez elementos (se os houbera) e en caso de pulsalo mostraríase un botón que permite volver aos dez elementos anteriores. Na figura 7.19 podemos observar o aspecto desta pantalla, mostrando tres lugares diferentes. O compoñente de React Native definido para isto é o chamado ”ShowPlaces” do directorio ”src/pages”.

RF-05 : Ver detalles dun lugar

Ao pulsar sobre un dos elementos da lista que se obtén ao realizar unha búsqueda, móstrase unha pantalla que contén os detalles do lugar sobre o que se realizou a acción. Esta pantalla, contén os atributos definidos para a entidade Location no modelo de datos ademais da funcionalidade de marcar o lugar como favorito pulsando sobre a estrela da parte superior dereita. Na figura 7.20 podemos observar o aspecto desta pantalla, mostrando os campos anteriormente mencionados, a estrela e máis un mapa coa localización do lugar. O compoñente de React Native definido para isto é o chamado ”LocationDetails” do directorio ”src/pages”.

RF-07 : Marcar como favorito

Para a funcionalidade de marcar un lugar como favorito créase un compoñente que consiste nunha estrela baleira por dentro que no momento que se pulsa sobre ela cambia o seu estado interno para converterse nunha estrela amarela e enviar ao servizo Rest unha petición para marcar o lugar indicado mediante props como favorito (ou desmarcalo se a estrela xa se

RF-05: Ver detalles dun lugar
Mostrar os detalles dun lugar concreto xunto coas puntuacións que os usuarios fixeron sobre el e as accións que sobre el se poden realizar
Tarefas
Deseñar e crear a vista para mostrar a lista de lugares resultados da búsqueda

Figura 7.15: Sprint 3: Ver detalles dun lugar

RF-07: Marcar como favorito
Marcar un lugar como favorito do usuario
Tarefas
Implementar a funcionalidade de marcar como favorito no servizo Rest
Deseñar o compoñente que permita lanzar a acción de marcar como favorito

Figura 7.16: Sprint 3: Marcar como favorito

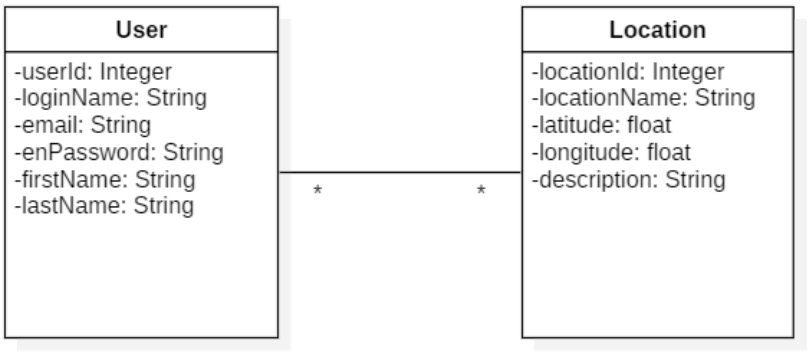

Figura 7.17: Modelo de datos do Sprint 3, unha vez engadida a entidade Location

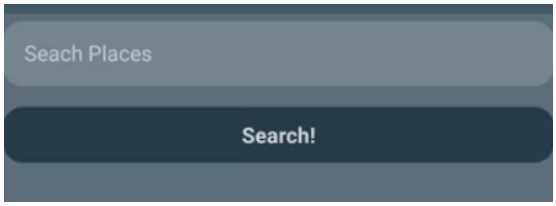

Figura 7.18: Vista da pantalla de búsqueda

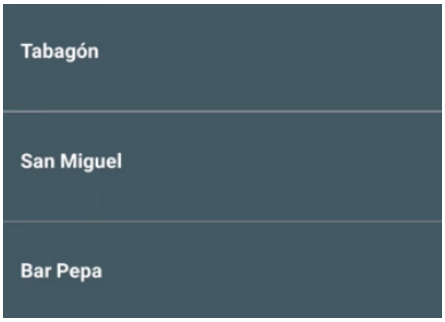

Figura 7.19: Vista do resultado dunha búsqueda de menos de dez elementos

Figura 7.20: Vista da pantalla que mostra os detalles dun lugar concreto

atopaba de cor amarelo). Na figura 7.20 podemos observar o aspecto deste compoñente antes de ser activado na parte superior dereita da pantalla. O compoñente de React Native definido para isto é o chamado ”Star” do directorio ”src/components”.

7.5.3 Implementación

A implementación deste Sprint non supuxo tanta complicación como a do Sprint anterior, pois as tarefas realizadas eran do mesmo tipo, polo que a práctica e o coñecemento sobre as tecnoloxías fixo que se realizasen sen ningún inconvinte.

Sen embargo existe unha cuestión que debe ser explicada, e é como se realiza o filtrado da búsqueda en función da distancia. Para realizar este tipo de tarefas relacionadas coa situación espacial e con distancias o recomendable é utilizar unha base de datos que dea soporte a este tipo de funcionalidades, como pode ser PostGIS [24]. Dado que na aplicación só se ía facer uso deste tipo de funcionalidades neste caso concreto decidiuse utilizar o xestor de base de datos PostgreSQL para evitar cambiar de xestor durante o desenvolvemento do proxecto.

Para solventar esta carencia o que se fixo foi definir unha función en PostgreSQL que en función de dous pares de coordenadas calcula a distancia que existe entre eles. Desta maneira, a hora de recuperar a información da base de datos e facilitando as coordenadas da posición actual pódense devolver soamente aqueles lugares que cumpren a condición de distancia imposta. A continuación podemos observar o aspecto que garda esta función.

```

create or replace function gc_dist(_lat1 float8, _lon1 float8, _lat2
float8, _lon2 float8) returns float8 as
$$
  select ACOS(SIN($1)*SIN($3)+COS($1)*COS($3)*COS($4-$2))*6371;
$$ language sql immutable;
```

7.6 Sprint 4: Comentarios

Neste Sprint levanse a cabo aquelas tarefas relacionadas cos comentarios que unha puntuación leva asociados tendo en conta que nun primeiro momento o requisito RF-06 : Comentar (despois RF-06: Puntuar) unicamente facía referencia ao acto de comentar un lugar, para máis tarde ser modificado no Product Backlog e incluír os comentarios como unha parte da acción de puntuar. De igual maneira, o requisito RF-20: Ver comentarios (despois RF-20: Ver puntuacións) que nun primeiro lugar só mostraba os comentarios realizados sobre un lugar en concreto, logo foi modificado cando os comentarios foron unha asociación das puntuacións realizadas polos usuarios. Para a consecución do Sprint escolléronse as entradas correspondentes do Product Backlog descompoñéndoas en tarefas para conseguir así o Sprint Backlog.

7.6.1 Análise

Nesta sección defínese o aspecto do Sprint Backlog unha vez escollidas as entradas e definidas as tarefas. Para este Sprint escolléronse as historias de usuario RF-06: Puntuar, RF-18: Modificar comentario, RF-19: Borrar puntuación e RF-20: Ver puntuacións que se poden ver descompostas nas figuras 7.21, 7.22, 7.23 e 7.24 respectivamente.

RF-06: Comentar (Agora RF-06: Puntuar)
Comentar un lugar aportando así unha opinión sobre a súa accesiblidade
Tarefas
Ampliar o deseño do modelo de datos para incluir a entidade ”Comment”
Implementar a funcionalidade de comentar no servizo Rest
Deseñar o compoñente que permita comentar un lugar concreto

Figura 7.21: Sprint 4: Comentar (Agora Puntuar)

RF-18: Modificar comentario
Modificar o texto dun comentario en concreto
Tarefas
Implementar a funcionalidade de modificar un comentario no servizo Rest
Deseñar o compoñente que permita modificar un comentario concreto

Figura 7.22: Sprint 4: Modificar comentario

RF-19: Borrar comentario (Agora RF-19: Borrar puntuación)
Eliminar un comentario sobre un lugar concreto
Tarefas
Implementar a funcionalidade de eliminar un comentario no servizo Rest
Deseñar o compoñente que permita eliminar un comentario sobre un lugar concreto

Figura 7.23: Sprint 4: Borrar comentario (Agora Borrar puntuación)

7.6.2 Deseño

Na anterior iteración deixábamos un modelo de datos que contiña as entidades ”User” e ”Location” relacionadas entre si. Para este Sprint, é necesario ampliar este modelo para así poder dar soporte a todas as funcionalidades relacionadas cos comentarios. Para isto, engadiuse unha nova entidade ”Comment” que se relaciona coa entidade ”User” para identificar

quen realizou o comentario e coa entidade ”Location” para saber sobre que lugar se realizou. Na figura 7.25 podemos ver o resultado do modelo de datos neste Sprint.

Para as tarefas deste Sprint é necesario engadir o repositorio que accede aos datos da táboa ”Comment” na base de datos, ademais do seu servizo e controlador.

RF-06: Comentar (Agora RF-06: Puntuar)

A hora de realizar o deseño que permite executar a funcionalidade de comentar optouse por algo sinxelo e intuitivo, un cadro de texto para escribir o corpo do comentario e un botón de ”Submit” que permite lanzar a acción de comentar.

Na figura 7.26 podemos observar o aspecto desta pantalla. Esta funcionalidade está incluída dentro da pantalla de ”LocationDetails” dentro do directorio ”src/pages”.

RF-18: Modificar comentario

Para ofrecer a funcionalidade de modificar un comentario optouse por utilizar un modal (cadro de diálogo emerxente que permite realizar accións sen cambiar ao usuario de pantalla). Na figura 7.27 podemos observar o modal que aparece na pantalla unha vez o usuario pulsou sobre o comentario da súa propiedade e escolleu a opción de modificar (icona con forma de lapis). Esta funcionalidade está incluída dentro da pantalla de ”LocationDetails” dentro do directorio ”src/pages”.

RF-19: Borrar comentario (Agora RF-19: Borrar puntuación)

De igual maneira que a funcionalidade de modificar un comentario, optouse por utilizar un modal para ofrecer a posibilidade ao usuario de eliminar o comentario que realizou previamente. Na figura 7.28 podemos observar o modal que aparece na pantalla unha vez o usuario pulsou sobre o comentario da súa propiedade e escolleu a opción de eliminar (icona con forma de papeleira). Esta funcionalidade está incluída dentro da pantalla de ”LocationDetails” dentro do directorio ”src/pages”.

RF-20: Ver comentarios (Agora RF-20: Ver puntuacións)

Para a realización desta funcionalidade optouse por utilizar unha lista de elementos igual a utilizada para listar as localizacións na historia de usuario RF-04: Mostrar Lugares 4.2.1. Na figura 7.29 podemos ver o aspecto desta lista cuns comentarios realizados sobre un lugar. Esta funcionalidade está incluída dentro da pantalla de ”LocationDetails” dentro do directorio ”src/pages”.

RF-20: Ver comentarios (Agora RF-20: Ver puntuacións)
Ver os comentarios realizados sobre un lugar concreto
Tarefas
Implementar a funcionalidade que permita recuperar os comentarios dun lugar no servizo Rest
Deseñar o compoñente que permita mostrar os comentarios dun lugar concreto

Figura 7.24: Sprint 4: Ver comentarios (Agora Ver puntuacións)

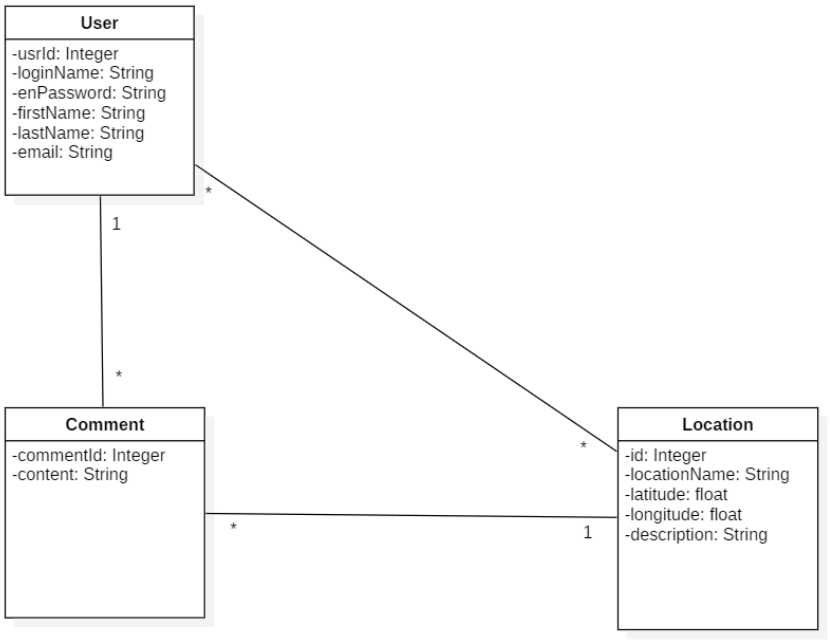

Figura 7.25: Modelo de datos do Sprint 4, unha vez engadida a entidade Comment

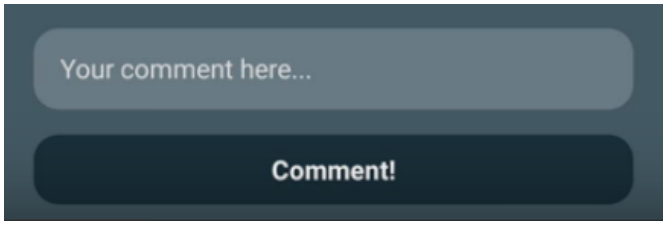

Figura 7.26: Vista dos detalles dun lugar na sección de comentar

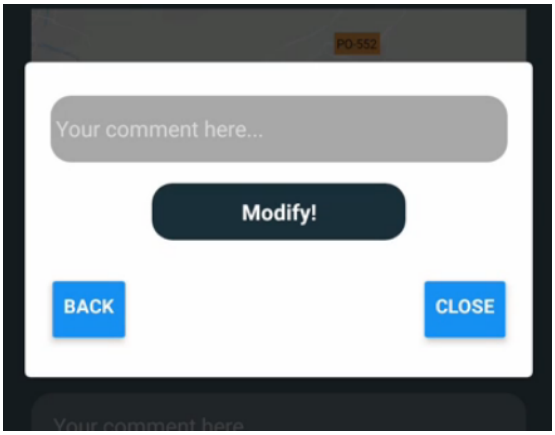

Figura 7.27: Vista do modal que permite a modificación dun comentario

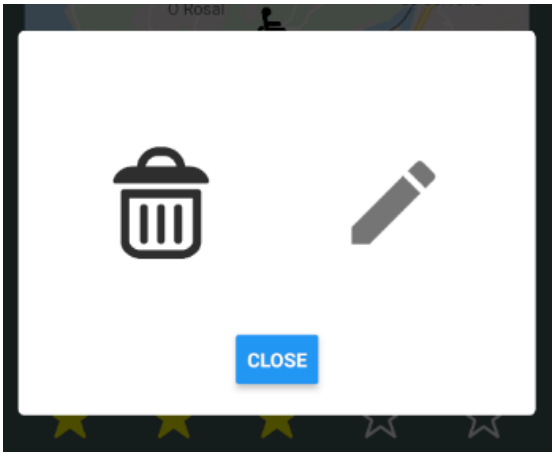

Figura 7.28: Vista do modal que permite a eliminación dun comentario

Figura 7.29: Vista da lista de comentarios realizados sobre un lugar

7.6.3 Implementación

A implementación deste Sprint non supón nada diferente aos outros dous anteriores, polo que non existe ningún aspecto que necesite ser explicado en profundidade pois a tarefa de implementación resultou ser algo mecánico.

7.7 Sprint 5: Puntuar e lugares favoritos

Neste Sprint lévanse a cabo as funcionalidades relacionadas coa puntuación dos lugares unha vez modificadas as entradas do Product Backlog referentes aos comentarios, pois a partir deste momento o comentario pasa a ser un contido asociado da puntuación pero mantendo o nome da entidade. Ademais disto, tamén se implementa a funcionalidade que permite recuperar e ver os lugares favoritos dun usuario. Para a consecución do Sprint escolléronse as entradas correspondentes do Product Backlog descompoñéndoas en tarefas para conseguir así o Sprint Backlog.

7.7.1 Análise

Nesta sección defínese o aspecto do Sprint Backlog unha vez escollidas as entradas e definidas as tarefas. Para este Sprint escolléronse as historias de usuario RF-06: Puntuar, RF-20: Ver puntuacións e RF-08: Ver lugares favoritos que se poden ver descompostas nas figuras 7.30, 7.31 e 7.32 respectivamente.

RF-06: Puntuar
Puntuar un lugar en función da accesibilidade que o usuario percibe
Tarefas
Modificar o modelo de datos para soportar as puntuacións
Modificar a funcionalidade de comentar para permitir puntuar
Modificar o compoñente que permitía comentar para poder puntuar

Figura 7.30: Sprint 5: Puntuar

7.7.2 Deseño

A nivel de deseño neste Sprint o único cambio necesario é engadir na entidade ”Comment” un novo atributo ”rating” que fai referencia a puntuación que cada usuario aporta a un lugar concreto en función da súa accesibilidade. Ademais disto, a media de puntuacións dun lugar (calculada automaticamente cando se puntúa) gárdase nun novo atributo chamado ”rating”

na entidade ”Location”. Na figura 7.33 podemos ver o resultado do modelo de datos neste Sprint.

RF-06: Puntuar

Para dar soporte a funcionalidade de puntuar, modifícase o deseño do compoñente que permite comentar sobre un lugar concreto. Para axudarnos a isto, inclúese un compoñente externo que ofrece un marcador para indicar a cantidade de estrelas que se desexa outorgar a un lugar. Na figura 7.34 podemos observar o cadro de texto para introducir o corpo do comentario, o marcador de ”rating” coas cinco estrelas e o botón de ”Submit” para realizar a acción de puntuar. Esta funcionalidade está incluída dentro da pantalla de ”LocationDetails” dentro do directorio ”src/pages”.

RF-20: Ver puntuacións

Para dar soporte a funcionalidade de ver as puntuacións, modifícase o deseño do compoñente que permite ver os comentarios dun lugar concreto. Para axudarnos nisto, utilízase o compoñente introducido para realizar as puntuacións na historia de usuario RF-06: Puntuar. Na figura 7.35 podemos observar unhas puntuacións realizadas sobre un lugar concreto no que se aprecia o corpo do comentario e a cantidade de estrelas que se lle outorgou a ese lugar. Esta funcionalidade está incluída dentro da pantalla de ”LocationDetails” dentro do directorio ”src/pages”.

RF-08: Ver favoritos

A hora de recuperar e mostrar os lugares favoritos dun usuario faese uso do compoñente utilizado na historia de usuario RF-04: Mostrar lugares 4.2.1, polo que é redundante volver a explicalo. A esta funcionalidade pódese acceder a través do compoñente ”MyProfile” situado no directorio ”src/pages” para despois dar paso ao compoñente ”ShowPlaces” situado na mesma carpeta.

7.7.3 Implementación

De igual maneira que no Sprint anterior, a implementación deste Sprint non supuxo nada diferente aos anteriores, polo que con respecto a implementación non existe ningún aspecto que necesite ser explicado.

7.8 Sprint 6: Peticións

Neste Sprint lévanse a cabo aquelas tarefas relacionadas coas peticións. Estas peticións, como xa se mencionou anteriormente poden ser de tres tipos: para engadir un novo lugar,

RF-20: Ver puntuacións
Ver as puntuacións realizadas sobre un lugar concreto xunto cos seus comentarios asociados (se os tiveran)
Tarefas
Modificar o compoñente para poder mostrar as puntuacións ademais dos comentarios

Figura 7.31: Sprint 5: Ver puntuacións

RF-08: Ver favoritos
Recuperar os lugares que o usuario marcou como favoritos
Tarefas
Implementar a funcionalidade que permita recuperar os lugares favoritos dun usuario no servizo Rest
Deseñar o compoñente que permita mostrar os lugares favoritos dun usuario

Figura 7.32: Sprint 5: Ver favoritos

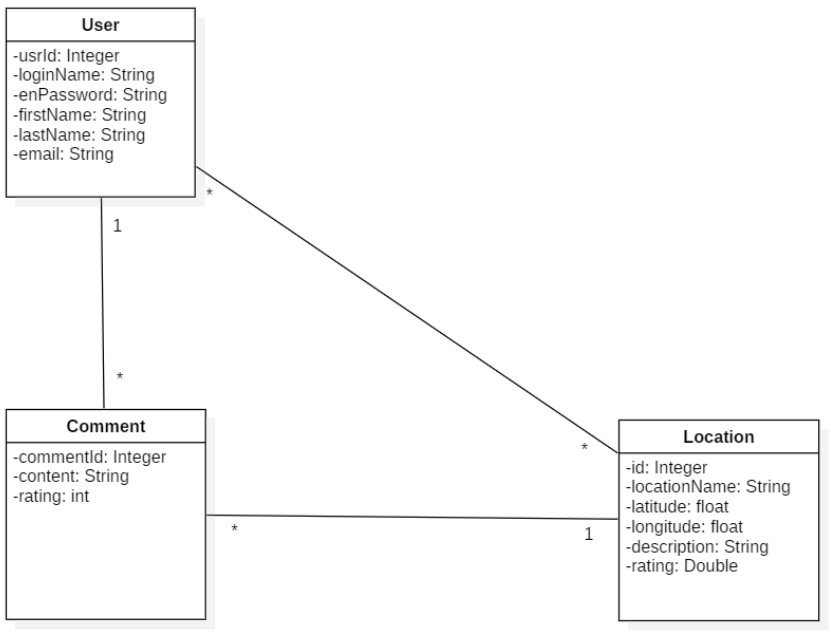

Figura 7.33: Modelo de datos do Sprint 5, agora con soporte para puntuar

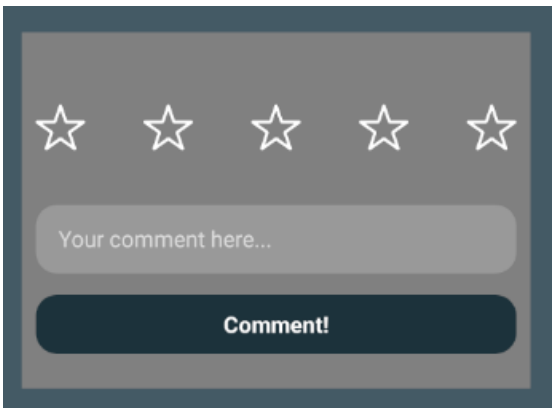

Figura 7.34: Vista dos detalles dun lugar no apartado de puntuar

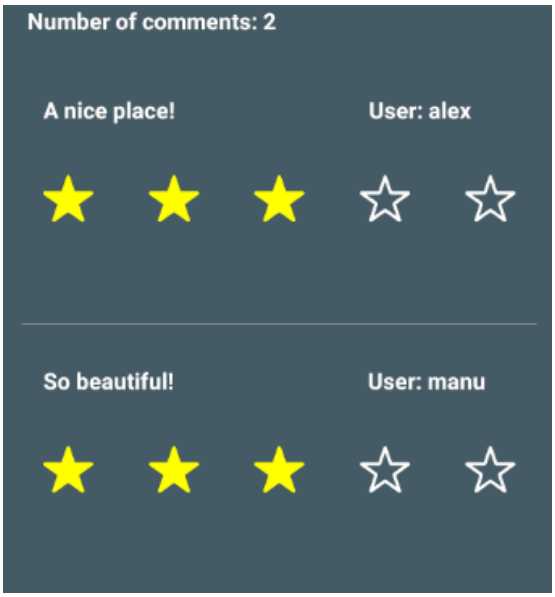

Figura 7.35: Vista das puntuacións realizadas sobre un lugar concreto

para borrar un lugar xa existente ou para modificalo. Para a consecución do Sprint escóllense as entradas correspondentes do Product Backlog descompoñéndoas en tarefas para conseguir así o Sprint Backlog.

7.8.1 Análise

Nesta sección defínese o aspecto do Sprint Backlog unha vez escollidas as entradas e definidas as tarefas. Para este Sprint escolléronse as historias de usuario RF-09: Engadir lugar, RF-10: Votar positivamente unha petición, RF-11: Votar negativamente unha petición, RF-12: Ver peticións de novos lugares, RF-13: Ver detalles dunha petición, RF-14: Realizar petición de borrado, RF-15: Ver peticións de borrado, RF-16: Realizar peticións de modificado e RF-17: Ver peticións de modificado que se poden ver descompostas nas figuras 7.36, 7.37, 7.38, 7.39, 7.40, 7.41, 7.42, 7.43 e 7.44 respectivamente.

RF-09: Engadir lugar
Realizar unha petición contra o sistema para engadir un novo lugar
Tarefas
Ampliar modelo de datos para incluir a entidade ”Petition”
Implementar a funcionalidade que permita facer unha petición para engadir un lugar no servizo Rest
Deseñar e implementar a pantalla que permita ao usuario realizar a petición contra a aplicación

Figura 7.36: Sprint 6: Engadir lugar

RF-10: Votar positivamente unha petición
Votar positivamente unha petición realizada para engadir un novo lugar ao sistema
Tarefas
Realizar a implementación da funcionalidade que permita votar positivamente unha petición
Deseñar e engadir un compoñente que permita votar positivamente na aplicación

Figura 7.37: Sprint 6: Votar positivamente unha petición

RF-11: Votar negativamente unha petición
Votar negativamente unha petición realizada para engadir un novo lugar ao sistema
Tarefas
Realizar a implementación da funcionalidade que permita votar negativamente unha petición
Deseñar e engadir un compoñente que permita votar negativamente na aplicación

Figura 7.38: Sprint 6: Votar negativamente unha petición

RF-12: Ver peticións de novos lugares
Ver a lista de peticións que os usuarios fixeron para engadir novos lugares ao sistema
Tarefas
Realizar a implementación no servizo Rest que permita recuperar a lista de peticións para engadir novos lugares que se atopen activas
Deseñar e implementar unha pantalla que permita mostrar as peticións para engadir novos lugares que se atopen activas

Figura 7.39: Sprint 6: Ver peticións de novos lugares

RF-13: Ver detalles dunha petición
Ver os detalles dunha petición concreta xunto coas accións que pode realizar o usuario sobre ela
Tarefas
Implementar a funcionalidade no servizo Rest que permita recuperar os detalles dunha petición concreta
Deseñar e engadir un compoñente que permita ver os detalles dunha petición concreta

Figura 7.40: Sprint 6: Ver detalles dunha petición

RF-14: Realizar petición de borrado
Realizar unha petición para que se borre un lugar concreto
Tarefas
Ampliar modelo de datos para incluir a entidade ”DeletePetition”
Implementar a funcionalidade que permita facer unha petición para eliminar un lugar no servizo Rest
Deseñar e engadir un compoñente que permita ao usuario realizar a petición contra a aplicación

Figura 7.41: Sprint 6: Realizar petición de borrado

RF-15: Ver peticións de borrado
Ver a lista de petitións de borrado de lugares que se atopan no sistema
Tarefas
Realizar a implementación no servizo Rest que permita recuperar a lista de peticións para eliminar lugares que se atopen activas
Deseñar e implementar unha pantalla que permita mostrar as peticións para eliminar lugares que se atopen activas

Figura 7.42: Sprint 6: Ver peticións de borrado

RF-16: Realizar peticións de modificado
Realizar unha petición para que se modifiquen certos campos dun lugar
Tarefas
Ampliar modelo de datos para incluir a entidade "ModifyPetition"
Implementar a funcionalidade que permita facer unha petición para modificar un lugar no servizo Rest
Deseñar e implementar unha pantalla que permita ao usuario realizar a petición contra a aplicación

Figura 7.43: Sprint 6: Realizar peticións de modificado

RF-17: Ver peticións de modificado
Ver a lista de petitións de modificado de lugares que se atopan no sistema
Tarefas
Realizar a implementación no servizo Rest que permita recuperar a lista de peticións para modificar lugares que se atopen activas
Deseñar e implementar unha pantalla que permita mostrar as peticións para modificar lugares que se atopen activas

Figura 7.44: Sprint 6: Ver peticións de modificado

7.8.2 Deseño

O modelo de datos desta iteración xa se corresponde co modelo de datos final da aplicación, polo que o seu aspecto será o indicado na figura 7.1, onde podemos apreciar os tres tipos de peticións: Petition, DeletePetition e ModifyPetition.

RF-09: Engadir lugar

Para dar soporte a funcionalidade de lanzar unha petición para engadir novos lugares na aplicación deseñouse unha pantalla cos campos necesarios que posúe a entidade ”Location” para poder cubrilos e a continuación lanzar a petición contra o sistema. Na figura 7.45 podemos observar a pantalla tal e como se acaba de describir, con diferentes campos de texto para cada un dos atributos da entidade ”Location”, un mapa interactivo para poder marcar as coordenadas do lugar e un compoñente chamado ”GPlaces” o cal coa axuda do motor de Google permítenos buscar lugares por palabras clave e desta maneira introducir as coordenadas máis facilmente. A pantalla que realiza esta funcionalidade é a de ”NewPlace” dentro do directorio ”src/pages”.

RF-10: Votar positivamente unha petición

Para dar soporte a funcionalidade de votar positivamente sobre unha petición concreta engádese un botón na parte inferior da pantalla referente aos detalles dunha petición. Na figura 7.46 podemos observar a pantalla mencionada e na parte inferior dereita o botón en cuestión. A pantalla que contén esta funcionalidade é a de ”PetitionDetails” dentro do directorio ”src/pages”.

RF-11: Votar negativamente unha petición

Para dar soporte a funcionalidade de votar negativamente sobre unha petición concreta engádese un botón na parte inferior da pantalla referente aos detalles dunha petición. Ao igual que na historia de usuario RF-10: Votar positivamente unha petición, podemos observar na figura 7.46 a pantalla mencionada e na parte inferior esquerda o botón en cuestión. A pantalla que contén esta funcionalidade é a de ”PetitionDetails” dentro do directorio ”src/pages”.

RF-12: Ver peticións de novos lugares

Para mostrar as peticións activas que fan referencia a adición de novos lugares realízase o deseño dunha pantalla que consiste nunha lista de elementos nos que se pode ler o nome da localización que tería en caso de aceptarse a petición para cada un dos elementos. En caso de ter máis de dez elementos aparecería na parte inferior da pantalla un botón que permite mostrar os seguintes dez elementos (se os houbera) e en caso de pulsalo mostraríase un botón que permite volver aos dez elementos anteriores. Na figura 7.47 podemos observar o aspecto

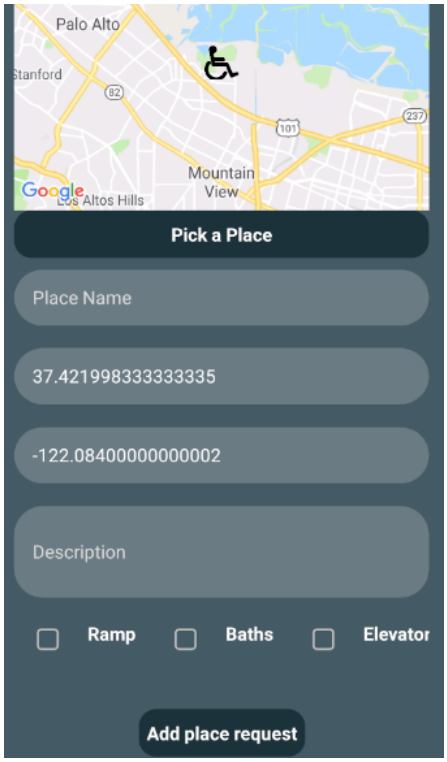

Figura 7.45: Vista da pantalla que permite realizar unha petición para engadir un lugar

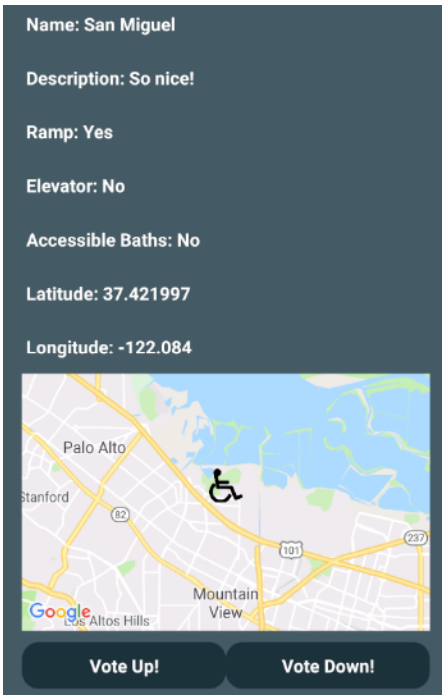

Figura 7.46: Vista da pantalla dos detalles dunha petición concreta

desta pantalla, mostrando varias peticións activas e na parte superior tres botóns que indican o tipo de peticións que se desexan mostrar, neste caso peticións de novos lugares. O compoñente de React Native definido para isto é o chamado ”ShowPetitions” do directorio ”src/pages”.

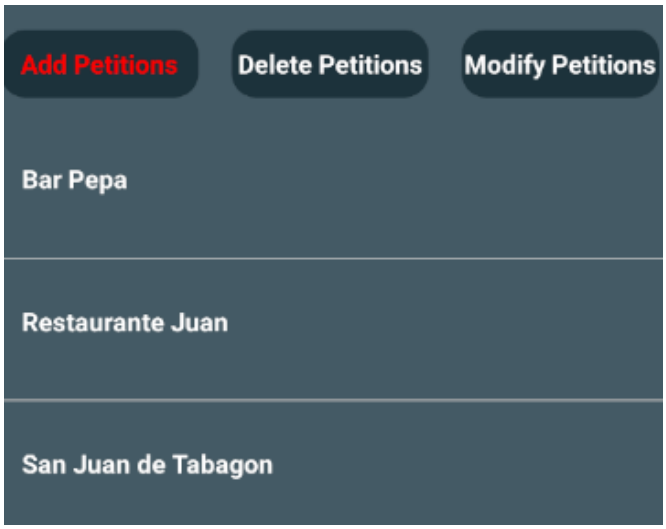

Figura 7.47: Vista da pantalla que mostra as peticións activas

RF-13: Ver detalles dunha petición

Ao pulsar sobre un dos elementos da lista que se obtén ao mostrar as peticións activas móstrase unha pantalla que contén os detalles da petición sobre a que se realizou a acción. Esta pantalla contén os atributos definidos para a entidade ”Petition” no modelo de datos ademais dos botóns que permiten votar sobre a petición e o contador de votos. Na figura 7.46 podemos observar o aspecto desta pantalla, mostrando os campos anteriormente mencionados, os botóns de votación e máis o contador. O compoñente de React Native definido para isto é o chamado ”PetitionDetails” do directorio ”src/pages”.

RF-14: Realizar petición de borrado

Para a funcionalidade de realizar unha petición de borrado sobre un lugar do sistema créase un compoñente que consiste nunha icona en forma de papeleira situada na pantalla de detalles dun lugar concreto. Ao pulsar sobre esta icona a aplicación mostra un modal no medio da pantalla que pregunta ao usuario se desexa realmente lanzar a petición de borrado. En caso de aceptar a petición será lanzada e o modal pecharase, en caso de cancelar o modal desaparecerá e non haberá efectos secundarios.

Na figura 7.49 podemos observar o aspecto deste compoñente antes de ser activado na parte superior dereita da pantalla e na figura 7.48 podemos ver o modal unha vez despregado cos botóns antes mencionados. O compoñente de React Native está situado na pantalla de ”LocationDetails” no directorio ”src/pages”.

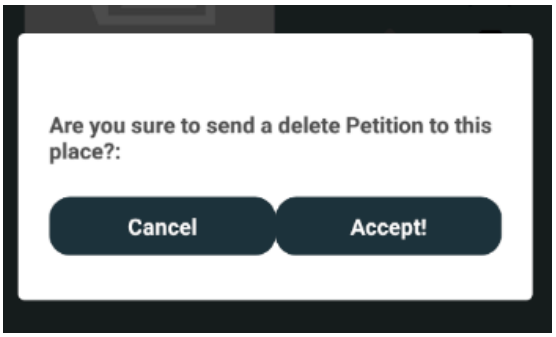

Figura 7.48: Vista do modal despregado para lanzar unha petición de borrado

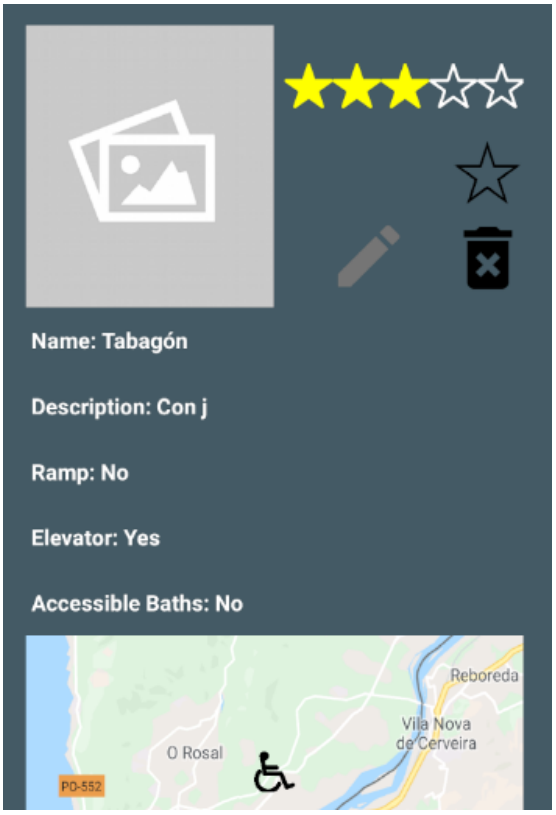

Figura 7.49: Vista da pantalla de detalles dunha localización (versión da aplicación acorde ao Sprint actual)

RF-15: Ver peticións de borrado

Para mostrar as peticións activas referentes as que teñen intención de eliminar un lugar rexistrado no sistema faese uso da mesma pantalla utilizada na historia de usuario RF-12: Ver peticións de novos lugares, que consiste nunha lista de elementos nos que en cada un se pode ler o nome da localización que se desexa borrar. En caso de ter máis de dez elementos aparecería na parte inferior da pantalla un botón que permite mostrar os seguintes dez elementos (se os houbera) e en caso de pulsalo mostraríase un botón que permite volver aos dez elementos anteriores.

Na figura 7.47 podemos observar o aspecto desta pantalla, mostrando varias peticións activas e na parte superior tres botóns que indican o tipo de peticións que se desexan mostrar. O compoñente de React Native definido para isto é o chamado ”ShowPetitions” do directorio ”src/pages”.

RF-16: Realizar peticións de modificado

Para a funcionalidade de realizar unha petición de modificado sobre un lugar do sistema créase un compoñente que consiste nunha icona en forma de lapis situada na pantalla de detalles dun lugar concreto. Ao pulsar sobre esta icona a aplicación redirixirá ao usuario a unha pantalla onde se poden observar os campos modificables dunha localización cubertos cos datos do lugar en cuestión.

Na figura 7.49 podemos observar o aspecto deste compoñente antes de ser activado na parte superior dereita da pantalla e na figura 7.50 podemos ver a pantalla que permite lanzar unha petición de modificado, cos campos que se permiten modificar dun lugar, checkboxs para indicar se ten rampa, ascensor e/ou baños adaptados e o botón que permite realizar a petición contra o servizo Rest. O compoñente de React Native está situado na pantalla de ”LocationDetails” no directorio ”src/pages” e unha vez accionado redirixe ao usuario a pantalla ”ModifyPetition” situada no directorio ”src/pages”.

RF-17: Ver peticións de modificado

Para mostrar as peticións activas referentes as que teñen intención de modificar un lugar rexistrado no sistema faese uso da mesma pantalla utilizada nas historias de usuario RF-12: Ver peticións de novos lugares e RF-15: Ver peticións de borrado, que consiste nunha lista de elementos nos que se pode ler para cada un deles o nome da localización que se desexa modificar. En caso de ter máis de dez elementos aparecería na parte inferior da pantalla un botón que permite mostrar os seguintes dez elementos (se os houbera) e en caso de pulsalo mostraríase un botón que permite volver aos dez elementos anteriores. Na figura 7.47 podemos observar o aspecto desta pantalla, mostrando varias peticións activas e na parte superior tres botóns que indican o tipo de peticións que se desexan mostrar. O compoñente de React Native definido para isto é o chamado ”ShowPetitions” do directorio ”src/pages”.

7.8.3 Implementación

No caso deste Sprint a implementación non foi algo mecánico, pois neste caso concreto é necesario buscar unha solución para o tratamento das peticións, é dicir, unha solución que permita comprobar se as peticións están activas ou non e en caso de haber rematado o seu período de vixencia comprobar se deben ser levadas a cabo.

Para conseguir isto, faese uso dunha utilidade de Spring, as ”Scheduling Tasks” [25] ou tarefas programadas. As tarefas programadas consisten nun código que se executa en función do tempo que ti lle indiques, tendo a posibilidade de executarse de maneira periódica, ao iniciar a aplicación, ao finalizar a súa execución, etc. A continuación podemos ver un exemplo de código dunha tarefa programada que cada sesenta segundos escribirá na consola ”Executando algo”.

```
  //Exemplo de tarefa programada

  //Defínese un executor de tarefas
ScheduledExecutorService executor =
  Executors.newSingleThreadScheduledExecutor();

  //Defínise a tarefa a executar
Runnable task = () -> {

      System.out.println("Executando algo");

};

  //A tarefa indicada executarase cada sesenta segundos unha vez
  pasados os dez primeiros.
ScheduledFuture future = executor.scheduleAtFixedRate(task, 10,
  60, TimeUnit.SECONDS);

future.get();
```

No noso caso, e para asegurarnos de que as peticións son atendidas incluso cando o servizo cae e volve ser posto en funcionamento, realízase a implementación das tarefas programadas a través dun EventListener chamado ApplicationReady que se lanza unha vez a aplicación está iniciada por completo. Dentro do método que trata este evento o que se fae é lanzar unha tarefa programada que cada ”x” tempo recupera todas as peticións activas e comproba se finalizaron ou non, e de ser así, executa as accións pertinentes en funcións dos votos de cada unha. Para poder utilizar as tarefas programadas é necesario facer uso da anotación @EnableScheduling que asegura que se crea un executor de tarefas en segundo plano.

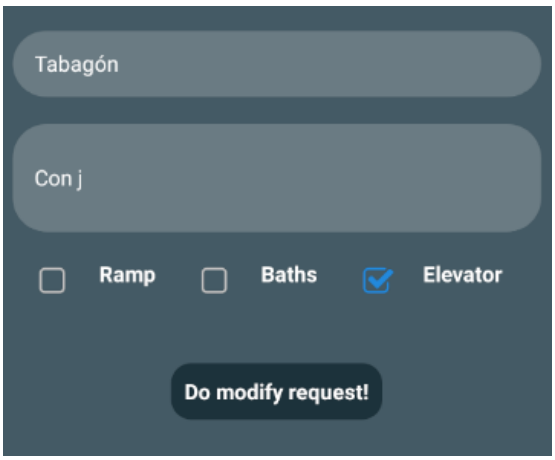

Figura 7.50: Vista da pantalla que permite lanzar unha petición de modificado sobre un lugar concreto

Capítulo 8

Conclusións e traballo futuro

8.1 Conclusións

Ao rematar o proxecto concluíuse que os obxectivos principais que se buscaban foron acadados.

En primeiro lugar a funcionalidade que se buscaba para a aplicación foi implementada por completo, incluso levando a cabo certos aspectos da aplicación que non se contemplaran nun primeiro momento.

Por outra parte, o alumno acadou un coñecemento e unha experiencia suficientemente amplas como para poder levar a cabo o desenvolvemento dunha aplicación móbil ou dun servizo Rest de maneira independente e sen axuda. Grazas ao proceso de desenvolvemento do proxecto, moitos dos coñecementos adquiridos durante o grao foron afianzados, permitindo así aplicar algúns conceptos que só se habían explicado na teoría ou dos cales só se aplicaran algunhas pinceladas. Por outra parte, a consecución do proxecto permitiu ao alumno aprender e mellorar no ámbito do Frontend, pois durante o grao este campo apenas se toca.

Ademais do coñecemento, realizar este proxecto axudou a obter unha concienciación maior coas persoas con diversidade funcional, pois para a realización de moitas funcionalidades da aplicación foi necesario informarse e falar con persoas que se atopan nesa situación, o que axudou a entender a súa situación e a empatizar con eles.

8.2 Traballo futuro

Este proxecto constitúe un comezo para a aplicación, a partir do cal os usuarios irían suxerindo cambios e novas funcionalidades para ir crecendo pouco a pouco e crear así unha gran comunidade.

Polo contrario, aínda que non haxa feedback do usuario, existen certas funcionalidades que se poderían engadir na aplicación e que serían de gran axuda, como por exemplo:

- Filtro de búsqueda por cidade.
- Ligazón a Google Maps.
- Imaxes nas Localizacións e Peticións.
- etc.

Fóra do ámbito tecnolóxico, unha tarefa que se realizará unha vez presentado o proxecto será a de pórse en contacto con diversas organizacións e fundacións que traten este tipo de proxectos para intentar polo en produción, e así, desta maneira, axudar a todas aquelas persoas que o precisen mediante o seu uso.

Apéndices

Apéndice A

Manual de intalación

A continuación defínese o proceso necesario para executar tanto o servizo Rest como a aplicación móbil realizadas no proxecto que describe esta memoria.

A.1 Software necesario

Para a execución do servizo Rest simplemente será necesario ter instalado o xestor de bases de datos PostgreSQL e ter configurada a base de datos que utilizará o servizo.

Polo contrario, para a execución da aplicación móbil non será tan sinxelo, pois necesitaremos o seguinte software:

- Node.js
- JDK
- Maven
- Android SDK (en caso de utilizar un emulador Android)

A.2 Instalación do Backend

Unha vez configurada a base de datos que utilizará o noso servizo deberemos iniciar o xestor para poder ter acceso a mesma. Unha vez feito isto, poderemos iniciar o noso servizo Rest de dúas maneiras diferentes:

a) Utilizar o comando: **mvn spring-boot:run** que a partir da súa execución se encarga de compilar e despregar a aplicación no servidor integrado de Spring Boot.

b) Compilar e despregar manualmente a aplicación, o que conleva realizar os seguintes pasos:

   1. Instalar un servidor de aplicacións.
   2. Executar o comando **mvn install** no directorio raíz do proxecto.
   3. Copiar o ficheiro "war" xerado na carpeta **webapps** do servidor.
   4. Configurar e iniciar o servidor.

A.3 Instalación do Frontend

En primeiro lugar, debemos descargar as dependencias necesarias incluídas no ficheiro package.json executando para isto o comando **npm install** na raíz do proxecto. A continuación temos varias opcións para executar a nosa aplicación:

1. Utilizando un emulador Android.
2. Mediante un dispositivo Android.
3. Utilizando un dispositivo iOS.

Neste caso e debido a que o servizo Rest é executado en local e que non dispoñemos dun dispositivo iOS, a aplicación foi executada e probada nun emulador Android. Unha vez executado o emulador Android, será necesario lanzar o comando **react-native run-android** o cal instala e executa a aplicación no emulador. O proceso para un dispositivo Android sería o mesmo, pero tendo en conta que debe estar conectado ao PC no que estemos executando o comando.

Para o caso de iOS deberemos asegurarnos de que o dispositivo se atopa conectado ao PC no que executaremos o comando **react-native run-ios**, o cal funciona de igual maneira que o utilizado para dispositivos Android.

Relación de Acrónimos

REST *REpresentational State Transfer.*

API *Application Programming Interface.*

POM *Project Object Model.*

NPM *Node Package Manager.*

XML *eXtensible Markup Language.*

JSON *JavaScript Object Notation.*

JSX *JavaScript XML.*

JWT *JSON Web Token.*

JVM *Java Virtual Machine.*

MVCC *Multiversion concurrency control.*

UWP *Universal Windows Platform.*

DOM *Document Object Model.*

CSS *Cascading Style Sheets.*

HTML *HyperText Markup Language .*

SCM *Software Configuration Management.*

HTTP *Hypertext Transfer Protocol.*

Glosario

Framework Esquema ou estrutura que se establece e que se utiliza para desenvolver e organizar un software determinado.

Webview Espazo no que se renderiza contido web.

Open-source Modelo de desenrolo de software basado na colaboración aberta entre usuarios.

Objective-c Lenguaxe de programación orientado a obxectos creado como un superconxunto de C.

Frontend Capa coa que interactúa un usuario nunha web ou aplicación.

Backend Capa que realiza a lóxica de negocio e o acceso a datos dun sistema.

Diagrama de Gantt Ferramenta gráfica cuxo obxectivo é expor o tempo de dedicación previsto para diferentes tarefas ou actividades ao largo dun tempo total determinado.

Props En React e React Native dise dunha serie de entradas arbitrarias que se pasan entre compoñentes.

Bibliografía

[1] "React native," https://facebook.github.io/react-native/.

[2] "Spring boot," https://spring.io/projects/spring-boot.

[3] "Scrum," https://www.scrum.org/resources/blog/que-es-scrum/.

[4] "Rest," https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html.

[5] "Java," https://docs.oracle.com/javase/7/docs/technotes/guides/language/.

[6] "Junit," https://junit.org/.

[7] "Maven," https://maven.apache.org/.

[8] "Postgresql," https://www.postgresql.org/.

[9] "Páxina oficial de javascript," https://www.javascript.com/.

[10] "Ecmascript," https://www.ecma-international.org/publications/standards/Ecma-262.htm.

[11] "React," https://es.reactjs.org/.

[12] "Redux," https://es.redux.js.org/.

[13] "Npm," https://www.npmjs.com/.

[14] "Nodejs," https://nodejs.org/.

[15] "Git," https://git-scm.com/.

[16] "Github," https://github.com/.

[17] "Sonarqube," https://www.sonarqube.org/.

[18] "Jenkins," https://jenkins.io/.

[19] "Redmine," https://www.redmine.org/.

[20] "Tutorial sobre javascript," https://javascript.info/.

[21] "Expo," https://expo.io/.

[22] "Api fetch javascript," https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API.

[23] "Páxina oficial de jwt," https://jwt.io/.

[24] "Postgis," https://postgis.net/.

[25] "Scheduling tasks," https://spring.io/guides/gs/scheduling-tasks/.