Facultade de Informática
UNIVERSIDADE DA CORUÑA

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

Aplicación web para unha empresa de arte funerario

Estudante: David García Gondell
Dirección: Fernando Bellas Permuy

A Coruña, xuño de 2021.

a miña familia e amigos polo apoio incondicional.

Agradecementos

Para min é primordial agradecer tanto aos meus pais como á miña familia e amigos polo apoio e o cariño que me fixo a persoa que son e que me permitiu cumprir coas miñas metas ata este momento. Especialmente agradezo os meus pais, que sempre me axudan e animan nos momentos difíciles.

Tamén quero agradecer a Fernando polas ideas, consellos, preocupación e paciencia que tivo durante a realización deste traballo.

Resumo

O obxectivo deste proxecto consiste no deseño e implementación dunha aplicación web inspirada nas necesidades dunha empresa familiar de venta de gravados. Esta aplicación debe ter a función de realizar pedidos pagándoos de forma online e debe permitir a un usuario administrador ter o control tanto dos pedidos como dos usuarios creados en dita aplicación.

A lóxica de negocio esta comprendida no backend, o cal consiste nunha API REST programada en Java xunto co framework Spring Boot. Este backend utiliza tamén MySQL para persistir os datos da aplicación.

O frontend da aplicación consiste nunha aplicación SPA desenvolta utilizando Javascript xunto cos frameworks React, Material-UI e a libraría Redux.

Abstract

The goal of this project is to design and implement a web application inspired by the needs of a family business which makes and sells prints. This application must have the functionality to make orders by paying them online and must allow an administrator to have the control over both orders and users created in the application.

The bussiness logic is included in the backend, which consists in a REST API programmed in Java along with Spring Boot. This backend also uses MySQL to persist the application's data.

The application's frontend consists of a SPA developed using Javascript along with the frameworks React, Material-UI and the library Redux.

Palabras chave:

- Backend
- Frontend
- Desenvolvemento
- Aplicación Web
- Spring Boot
- React
- Redux
- Gravados

Keywords:

- Backend
- Frontend
- Development
- Web Application
- Spring Boot
- React
- Redux
- Prints

Índice Xeral

Índice de Figuras

Índice de Táboas

Capítulo 1

Introdución

Na actualidade, xa ninguén pode imaxinar a nosa vida sen os avances que nos ofrece internet. Isto aínda se fai mais evidente no mundo laboral no que nas empresas teñen que competir pola clientela atraendo á maior cantidade de xente posible para aumentar as ventas. A orixe da idea para este proxecto ven precisamente deste mundo empresarial.

O problema comeza cunha empresa familiar de gravados. Esta empresa ten á sua disposición unha tenda na que se venden placas de cemiterio gravadas e unha nave industrial na que están as ferramentas necesarias para a realización de ditas placas. Nela se recibe unha gran cantidade de pedidos pero estes non son recibidos polos traballadores da nave industrial ata que se poñen en contacto mediante teléfono.

Como consecuencia, é necesario facer chamadas periódicamente para pasar os pedidos que se reciben na tenda, entorpecendo aínda que non sexa de forma grave á fabricación das placas de cemiterio.

Además a empresa dispón dunha páxina web, pero esta anticuada e actualmente non ten ningunha forma de realizar os pedidos online, polo que nela teñen un teléfono e email de contacto para facer pedidos. Por este motivo os clientes vense obligados a mirar o catálogo dispoñible en forma de PDF na páxina web, escoller a placa que queren e poñer toda a súa especificación por email ou explicala por teléfono, o cal pode ser tedioso para ditos clientes.

Por todo isto, unha aplicación web pode ser a solución máis axeitada para esta empresa. Esta aplicación web sustituiría a páxina que teñen actualmente, permitindo aos clientes facer os pedidos online ao mesmo tempo que renovan a páxina web e obteñen unha lista dos pedidos realizados na aplicación dende calquer dispositivo con acceso a internet.

1.1 Obxectivos

O obxectivo do proxecto é crear unha aplicación web que terá como propósito a venta online de arte funeraria, é dicir, de placas de cemiterio e será posible a venta tanto a particulares

como a empresas.

A aplicación poderá soportar 3 tipos distintos de usuarios: particulares, empresas cliente e administradores. Os particulares son clientes individuais que poden rexistrarse na aplicación pola súa conta e suponse que encargan placas de forma esporádica. As empresas son clientes habituais (e.g. unha funeraria) que necesitan ser rexistrados na aplicación por un administrador e poden ter un desconto asociado. Os usuarios de tipo administrador son os correspondentes aos traballadores da empresa de arte funerario e se encargan de xestionar a información dos pedidos e do rexistro de usuarios da aplicación.

Unha vez se entre na aplicación, estará dispoñible para os clientes (particulares e empresas) un catálogo con todos os distintos prototipos de placas de cemiterio e poderase seleccionar un. Logo poderanse especificar os parámetros dunha placa de cemiterio para facer un pedido. Unha vez seleccionados todos os parámetros do pedido, este se poderá confirmar, o que no caso de ser un cliente empresa fará que se proceda ao pago do pedido de forma online, mentres que en caso contrario o pedido se engade á aplicación pendente de pago. Ademais, poderase acceder a un historial dos pedidos feitos polo usuario. Neste historial poderanse pagar os pedidos feitos por usuarios particulares que aínda estén pendentes de pago.

Por último, na sección dedicada ao usuario administrador poderanse rexistrar novos usuarios, así como acceder a unha lista dos mesmos. Tamén se poderá ver un listado de todos os pedidos e para permitir máxima flexibilidade, poderanse engadir, modificar e eliminar os mesmos.

1.2 Visión global do sistema

O sistema consistirá nunha aplicación web cun backend encargado da lóxica de negocio e un frontend encargado de xestionar a vista da mesma (figura 1.1).

O backend está implementado en Java, usa MySQL para persistir os datos e ofrece unha API REST. O frontend consiste nunha aplicación web moderna (SPA) implementada con JavaScript.

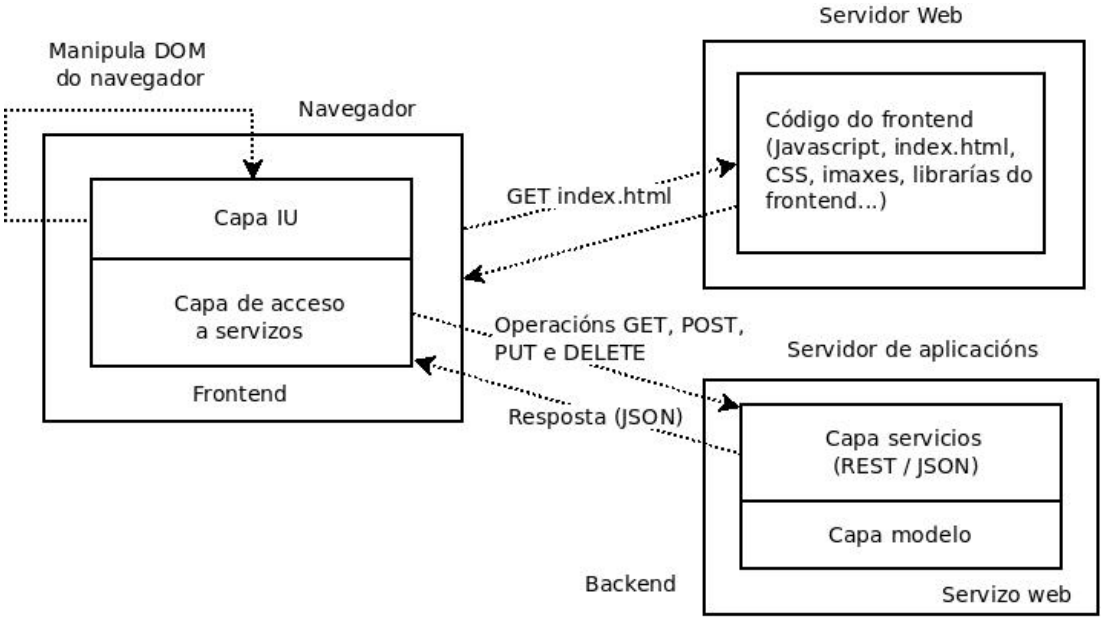

Figura 1.1: Diagrama da arquitectura da aplicación

Capítulo 2

Estado da arte

Na actualidade existen varias páxinas e aplicacións web que se dedican á venta de placas gravadas de cemiterio, pero non se atopu ningunha que tivera as mesmas funcionalidades que as que se abordan neste proxecto. A pesar disto existen aplicacións con servizos semellantes que inspiraron moitas funcionalidades que se atopan na aplicación desenvolta. A continuación se expoñen as máis relevantes.

2.1 LasPlacas.com

LasPlacas.com [1] é unha tenda de comercio electrónico que se dedica á venta de placas gravadas de calquera tipo, como industria, hostelería, comercios e moitos outros. Nesta tenda pódese escoller entre un gran número de placas nun catálogo, facer pedidos de placas de cemiterio e pagalas diretamente online mediante PayPal. Non obstante, esta páxina admite pouca personalización e só admite cambiar texto e tamaño da placa de cemiterio. A continuación móstrase unha imaxe do catálogo desta tenda online (figura 2.1).

Figura 2.1: Catálogo de placas na tenda LasPlacas.com.

Adicionalmente, esta páxina web tamén ofrece recomendacións dos productos máis destacados nunha barra lateral, reduccións dos prezos cando o importe supera un límite e recomendacións de productos por parte dos usuarios.

2.2 Rotumax

Rotumax [2] é unha aplicación web de venta de placas gravadas de todo tipo, igual que a tenda da sección anterior. A principal diferencia e o que fai a esta aplicación tan interesante é que dispón dun editor co cal personalizar as placas. Desta forma pódese escoller o material, fixacións, tamaño, imaxes e outros, además de poder escoller o lugar exacto onde se vai

mostrar o texto e os diferentes compoñentes da placa. Máis adiante amósase unha imaxe da personalización dunha placa nesta páxina (figura 2.2).

Figura 2.2: Personalización dunha placa en Rotumax

2.3 Taller de arte Renaud Gravure

Unha das aplicacións web que máis se semella á deste proxecto é o taller de arte Renaud Gravure [3], xa que esta especializada en arte funerario e permite seleccionar os distintos elementos da placa de cemiterio. Da mesma forma que Rotumax, tamén dispón dun editor co que escoller o lugar onde se colocan os distintos compoñentes da placa de cemiterio, pero ao estar máis especializada dispón de debuxos predefinidos máis axeitados e máis opcións entre as que escoller para personalizar ditas placas. A continuación amósase a personalización dunha placa de cemiterio na figura 2.3

Figura 2.3: Personalización dunha placa no taller de arte Renaud Gravure.

Esta, xunto con Rotumax é a páxina que máis personalización admite para as placas que vende, pero para acotar o tamaño deste proxecto optouse finalmente por non permitir ao usuario cambiar o lugar dos compoñentes da placa na aplicación web desenvolta.

2.4 Outras páxinas web de venta de arte funerario

Ademáis das xa citadas, existen moitas outras páxinas web dedicadas á venta de arte funerario, tanto con posibilidade de compra online como sen ela. Estas últimas aportan un teléfono ou email de contacto para facer os pedidos e por isto non son de grande interese para este proxecto. Algúns exemplos son:

- Grabinco [4]
- Trofeos Cadenas [5]
- lapidasvimianzo.com [6]
- placasgrabadas.es [7]

Capítulo 3

Metodoloxía

Neste capítulo da memoria explicarase a metodoloxía utilizada para levar a cabo o proxecto e as vantaxes e inconvenientes da mesma.

3.1 A metodoloxía seleccionada

Neste proxecto utílizase unha metodoloxía iterativa. Unha vez recollidos os requisitos globais da aplicación a desenvolver, realízanse varias iteracións nas que se leva a cabo un subconxunto do total dos casos de uso.

En cada unha das iteracións abordaranse as fases de análise, deseño, desenvolvemento e probas para o subconxunto de funcionalidades seleccionado en ditas iteracións. Unha vez rematen todas as fases o resultado dunha iteración debe ser unha aplicación web executable e que poida ser posta en funcionamento nese momento. Desta forma, ao remate de cada iteración a aplicación web será mais completa que ao remate da iteración anterior.

Este tipo de metodoloxía recorda en certa forma a SCRUM. Non obstante, neste proxecto que ten un equipo de desenvolvemento composto por unha soa persoa, non é posible utilizar esta metodoloxía xa que ten como unha das súas bases un equipo de desenvolvemento con distintos roles ben diferenciados como poden ser o ”Product Owner” ou o ”Scrum Master”. Non obstante, a metodoloxía utilizada neste proxecto si que esta fortemente inspirada nalgúns nos aspectos de SCRUM, xa que se pretende obter unha metodoloxía áxil. Isto permite adaptarse facilmente aos cambios que se poidan surxir durante a realización do proxecto e da a posibilidade de facer retrospectiva e mellorar aspectos do desenvolvemento ao longo das iteracións.

3.2 Vantaxes e inconvenientes

A continuación se expoñen algunhas das vantaxes e inconvenientes da utilización desta metodoloxía no proxecto. As vantaxes do desenvolvemento cunha metodoloxía iterativa como esta son:

- Os usuarios non necesitan esperar ao remate do proxecto para utilizar a aplicación.
- As funcionalidades de mais valor poden ser realizadas nas primeiras iteracións e estar dispoñibles antes.
- Flexibilidade ante cambios
- Existe a posibilidade de facer retrospectiva para mellorar aspectos do desenvolvemento do proxecto.

Algúns dos inconvenientes que pode presentar dita metodoloxía son:

- Pode ser complexo definir o núcleo operativo para lograr o primeiro incremento.
- Pode resultar complicado determinar as funcionalidades principais ou con máis valor para priorizar o seu desenvolvemento.
- É posible que unha solución para unha iteración non sexa valida en iteracións posteriores.

Con todo, aínda que a metodoloxía iterativa que se utiliza presente estes importantes inconvenientes, como se verá a continuación, a aplicación web a desenvolver ten unha sección de administración cunha prioridade clara e non existen aspectos que se necesiten nunha iteración pero non sexan válidos noutra. Por isto e polas grandes vantaxes que presenta considero esta metodoloxía axeitada para o proxecto a desenvolver.

Capítulo 4

Análisis de requisitos global

Neste capítulo descríbense os distintos roles de usuario que terán acceso a aplicación web e os requisitos globais da mesma.

4.1 Roles de usuario

Os distintos tipos de usuario dispoñibles na aplicación teñen como diferencias principais os permisos para entrar nas distintas partes da aplicación e a forma na que realizan novos pedidos de placas de cemiterio. Tendo en conta isto, a aplicación web consta de catro roles ben diferenciados:

- **Usuario sen autenticar:** os usuarios que non esten autenticados na páxina web só poderán acceder ao catálogo de placas de cemiterio e ao rexistro de usuarios particulares.
- **Administrador:** os administradores da páxina teñen acceso a unha sección específica da aplicación web na cal poden controlar todos os pedidos e usuarios rexistrados na mesma, así como engadir novos.
- **Usuario particular:** son clientes non habituais que entran na aplicación e se rexistran eles mesmos. Este tipo de usuario pode facer pedidos de placas de cemiterio pero non pode pagalos mentres non se lles asigne un bosquexo (se un pedido non esta pagado non comeza a facerse a súa placa correspondente).
- **Usuario cliente:** son empresas como poden ser funerarias que fan pedidos habitualmente. Este tipo de usuario é rexistrado na aplicación sempre por un administrador e cando fan pedidos os pagan no mesmo momento da súa creación. Ademais, os usuarios cliente poden ter un desconto aplicado sobre os pedidos que realicen.

4.2 Casos de uso

A continuación amósanse todos os casos de uso da aplicación explicados brevemente. Cabe destacar que as imaxes dos mockup non se corresponden totalmente coa versión final da aplicación ao estar feitas nunha fase moito mais temprana do desenvolvemento do proxecto.

CU1 - Autenticación de usuario: Na aplicación web poderán autenticarse todos os usuarios independentemente do seu rol, indicando o seu nome de usuario e contrasinal. En caso de que estos sexan incorrectos aparece unha mensaxe de erro.

CU2 - Rexistro de usuario particular: Os usuarios que entren na aplicación web poderán rexistrarse. Para isto é necesario introducir un nome de usuario, contrasinal, nome, apelidos, teléfono e email. Unha vez feito rexistraráse un usuario co rol de usuario particular na aplicación e quedara autenticado na mesma.

CU3 - Pechar sesión: Un usuario autenticado poderá pechar sesión, co que as súas posibilidades quedarán restrinxidas ás dun usuario sen autenticar.

CU4 - Mostrar catálogo de placas: Calquer tipo de usuario, xa sexa autenticado ou sen autenticar poderá ver o catálogo de placas. Este consistirá en imaxes de cada prototipo de placa de cemiterio dispoñible.

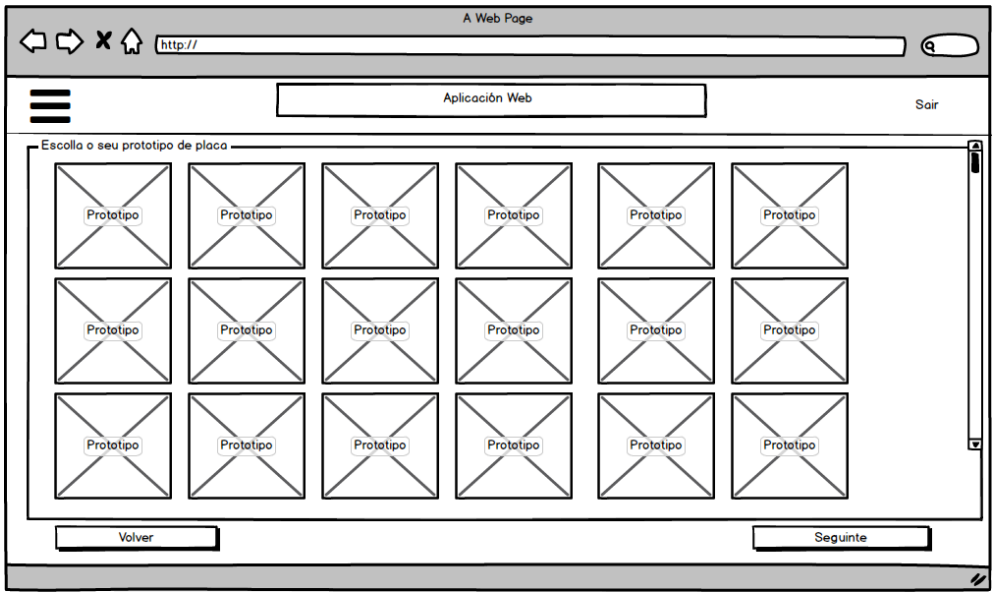

Figura 4.1: Mockup do catálogo de placas

CU5 - Facer pedidos de placas de cemiterio: Dende o catálogo de pedidos poderáse escoller un para facer un pedido. Unha vez feito, poderánse escoller os distintos compoñentes e parámetros da placa, que son:

- Material
- Tamaño
- Acabado
- Texto da placa
- Tipografía
- Cruz
- Debuxo
- Floreiro
- Observacións

Cando se escollen todos os parámetros anteriores o usuario pasará por unha confirmación de pedido na que se mostrarán todos eles. Logo, se é un usuario cliente o que fai o pedido procederá ao pago do mesmo, mentres que se é un usuario particular engadirase o pedido como non pagado.

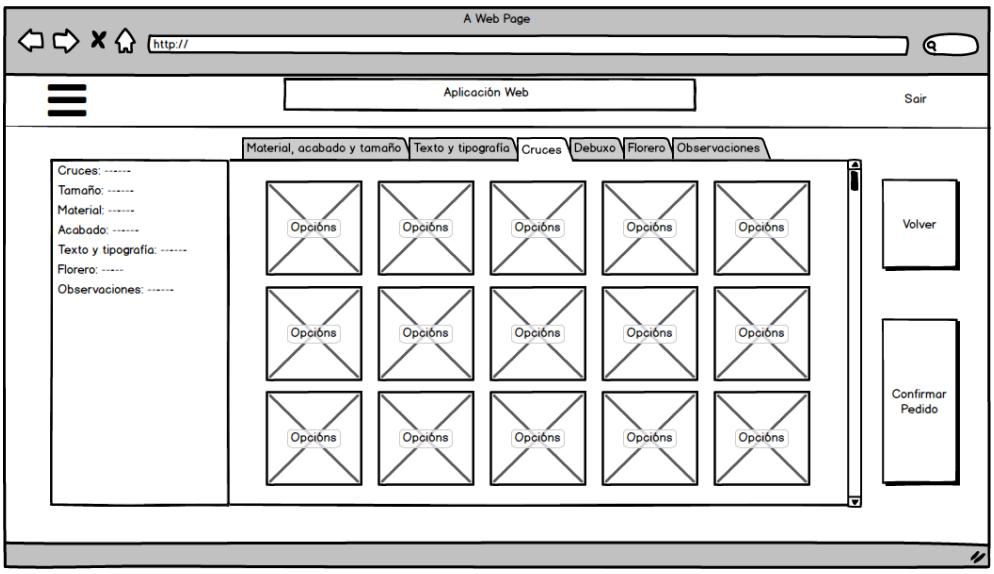

Figura 4.2: Mockup da selección dos parámetros dun pedido

CU6 - Cancelar pedido de placa de cemiterio: Un usuario poderá cancelar os seus pedidos sempre que estos non estén xa pagados.

CU7 - Ver bosquexo de pedido: Un particular poderá ver o bosquexo dun pedido. Este bosquexo sera engadido previamente por un administrador.

CU8 - Confirmar pedido e pagar: Se un pedido feito por un particular ten un bosquexo engadido, este usuario poderá pagalo de forma online mediante Paypal co que quedará rexistrado dito pedido como pagado.

CU9 - Mostrar pedidos dispoñibles: Un usuario calquera poderá acceder a unha lista cos pedidos feitos por el mesmo, mostrando para cada pedido o seu estado, a data na que se fixo o pedido e o seu prezo.

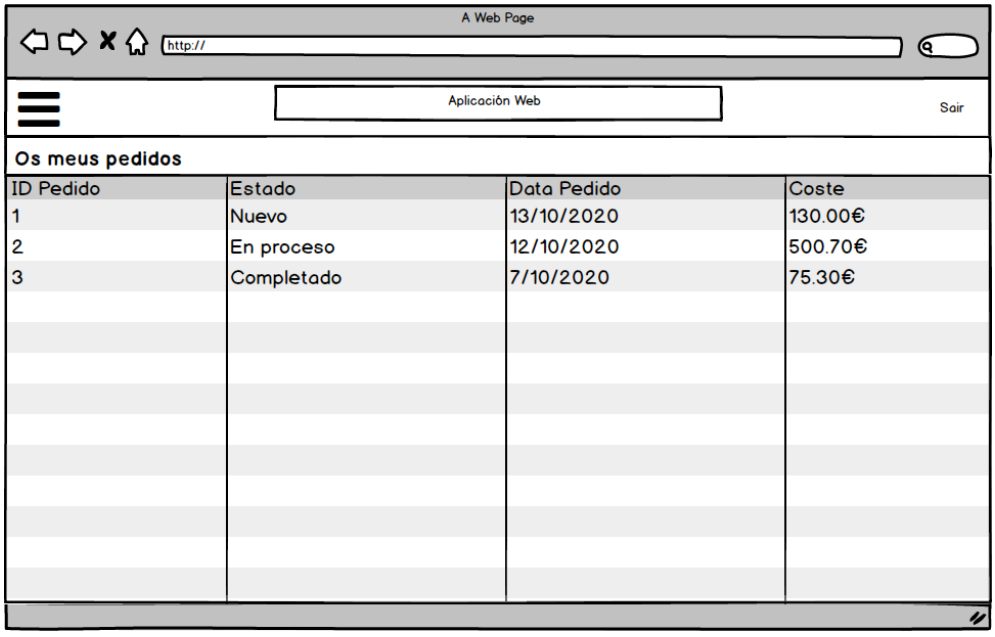

Figura 4.3: Mockup dos pedidos dun usuario

CU10 - Engadir usuarios: Un administrador poderá engadir outros usuarios de calquer tipo dispoñible na aplicación web.

CU11 - Mostrar historial de pedidos: Os usuarios administradores terán acceso a unha lista de todos os pedidos coa posibilidade de filtrar por usuario, data e estado.

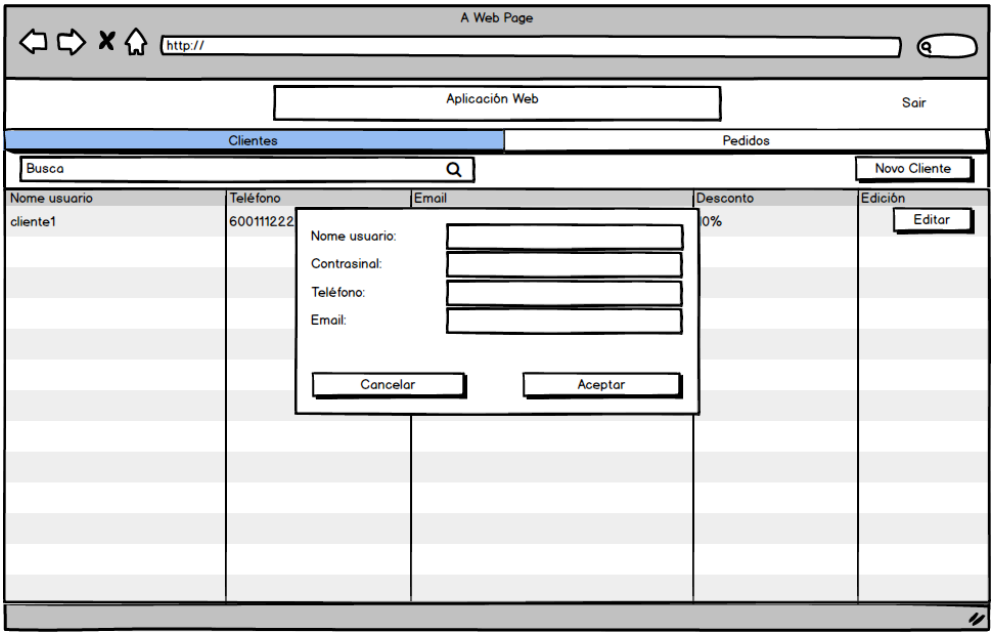

Figura 4.4: Mockup do engadido dun usuario

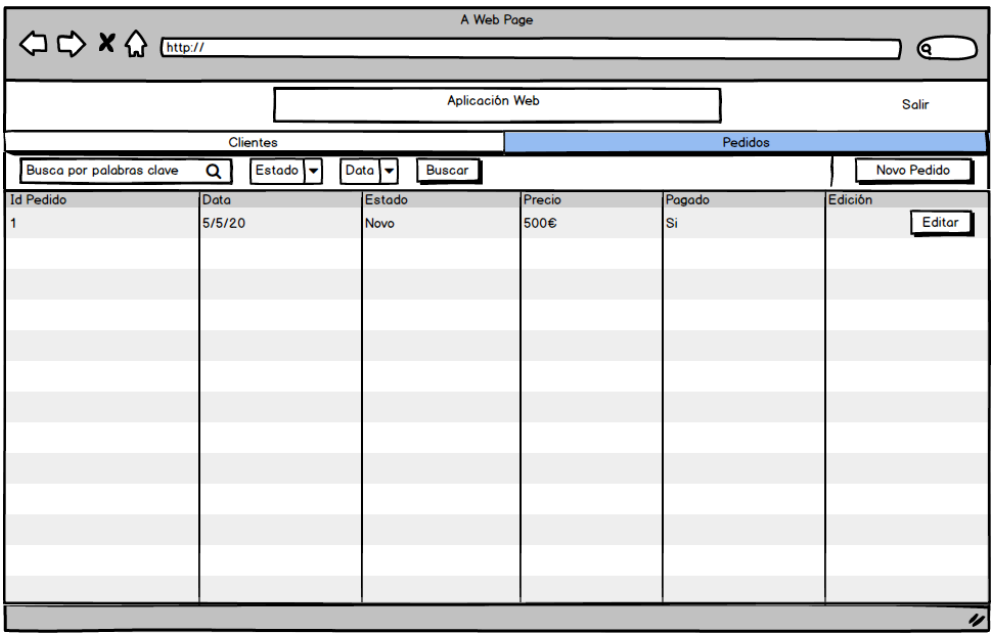

Figura 4.5: Mockup do historial de pedidos

CU12 - Engadir pedido: Os usuarios administradores terán a posibilidade de engadir un pedido indicando os distintos parámetros da placa.

CU13 - Cambiar estado de pedido: Un administrador poderá cambiar o estado dun pedido. Un pedido pode estar nos estados de novo, en proceso ou completado.

CU14 - Engadir bosquexo: Dende a sección de administración poderáse engadir un bosquexo a un pedido feito por un particular.

CU15 - Actualizar bosquexo: Un administrador poderá actualizar o bosquexo dun pedido.

CU16 - Actualizar información de pedido: Un administrador poderá cambiar toda a información dun pedido, xa sexan parámetros da placa encargada, o prezo do pedido ou outros.

CU17 - Eliminar pedido: Un usuario administrador poderá eliminar un pedido da aplicación.

CU18 - Mostrar Usuarios: Os administradores terán acceso a unha lista con todos os usuarios rexistrados na aplicación.

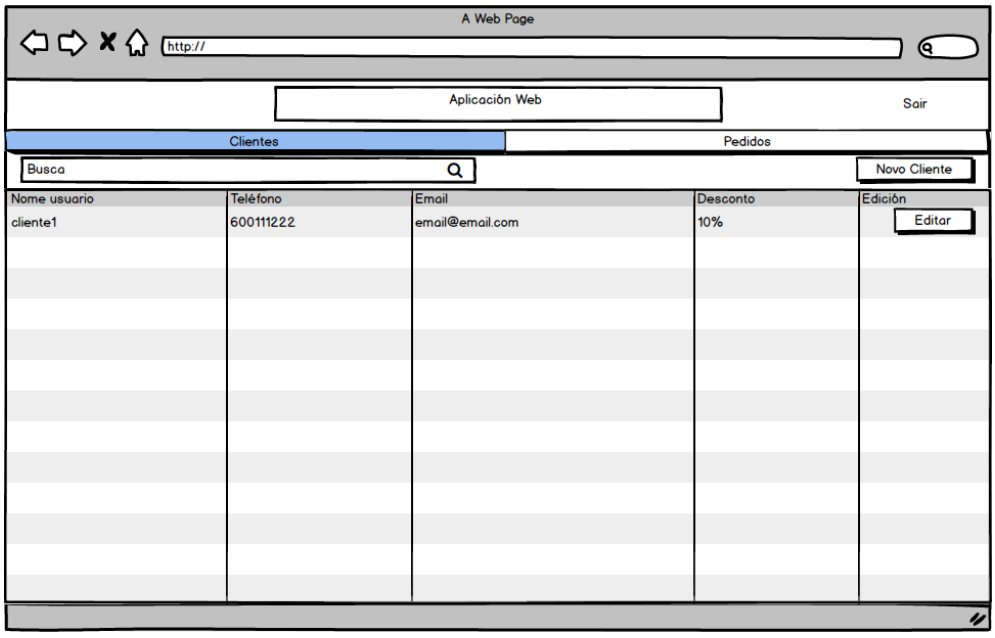

Figura 4.6: Mockup da lista de usuarios

CU19 - Aplicar desconto: Un administrador poderá cambiar o desconto dos usuarios.

CU20 - Modificar usuario Un administrador poderá cambiar a información relativa a un usuario.

CU21 - Eliminar usuarios: Dende a sección de administración poderase eliminar un cliente da aplicación web.

Capítulo 5

Planificación

No capítulo de planificación describirase como se levou a cabo a planificación do proxecto seguindo unha metodoloxía iterativa e os obxectivos de cada unha das iteracións abordadas ao longo do proxecto.

Tamén se exporá a planificación temporal do proxecto.

5.1 As iteracións

O proxecto, como se mencionou anteriormente segue unha metodoloxía iterativa, na que as funcionalidades da aplicación se reparten en varias iteracións. Estas teñen un obxectivo e deben rematar cunha aplicación que poida ser utilizada polo usuario final.

Este proxecto dividíuse en seis iteracións que se detallan a continuación:

5.1.1 Iteración 0

Antes de comezar o desenvolvemento foi necesaria unha iteración de análisis global de requisitos na que se decidiron os casos de uso que se levarían a cabo e fixéronse mockups correspondentes as distintas pantallas da aplicación web a desenvolver.

5.1.2 Iteración 1

O obxectivo da primeira iteración do proxecto na que se desenvolveron distintos casos de uso da aplicación consistíu na implementación da sección de administración da aplicación. Os casos de uso seleccionados para esta primerira iteración son os seguintes:

- CU1 - Login
- CU3 - Pechar sesión
- CU10 - Rexistro de usuarios cliente (empresas)

- CU11 - Mostrar o historial de pedidos
- CU12 - Engadir pedido
- CU13 - Cambiar estado de pedido
- CU16 - Actualizar información de pedido
- CU17 - Eliminar pedido
- CU18 - Mostrar usuarios
- CU19 - Aplicar desconto
- CU20 - Modificar usuario
- CU21 - Eliminar usuario
- CU22 - Detalle de pedido

5.1.3 Iteración 2

Na segunda iteración pretendíase desenvolver a sección de catálogo e realización de pedidos. Además, para a realización de pedidos dos usuarios clientes, tamén foi necesario desenvolver unha forma de pagamento online. Os casos de uso a abordar nesta iteración foron os seguintes:

- CU2 - Rexistro de usuarios particulares
- CU4 - Mostrar catálogo de placas
- CU5 - Facer pedido de placas de cemiterio

5.1.4 Iteración 3

Nesta iteración o obxectivo era o engadido dun historial de pedidos para os clientes, a posibilidade de poder ver o boceto dun pedido e pagar para os usuarios particulares. Neste caso abórdanse os casos de uso restantes:

- CU6 - Cancelar pedido de placa de cemiterio
- CU7 - Ver boceto de pedido
- CU8 - Confirmar pedido e pagar
- CU9 - Mostrar pedidos de usuario

- CU14 - Engadir boceto

- CU15 - Actualizar boceto

5.1.5 Iteración 4

Na cuarta iteración, o obxectivo non era o desenvolvemento de máis casos de uso, senón que se pretendía pulir as funcionalidades desenvoltas durante as anteriores tres iteracións, polo tanto, nesta non se abordou ningún caso de uso.

5.1.6 Iteración 5

A quinta e última iteración dedicouse a redacción desta memoria.

5.2 Planificación temporal

A planificación temporal das iteracións do proxecto fíxose de forma independente para cada unha ao seu comezo. Con respecto a duración das mesmas, pódese observar que a maior parte das horas empregadas recaeron na primeira e segunda iteración. Isto débese a que nestas iteracións era necesario o aprendizaxe do framework Material-UI na primeira e a API REST de pagamentos de Paypal na segunda. A pesar disto, na primeira iteración sobreestimouse a duración, xa que Material-UI axilizou moito o desenvolvemento das vistas da aplicación. Na segunda iteración como é máis habitual ao tratar con tecnoloxías novas, subestimouse moito a duración por non poder coñecer con exactitude canto tempo era necesario para poñer en funcionamento a API REST de pagamentos de Paypal e porque a documentación era extensa e mal organizada.

A continuación amósase unha táboa (5.1)que mostra a duración das iteracións do proxecto:

Sprint	Obxectivo	Data inicio	Data fin	Horas estimadas	Horas empregadas
0	Concepción do produto	12/11/19	10/11/19	-	30
1	Sección de administración	5/7/20	20/8/20	230	180
2	Catálogo e realización de pedidos	20/8/20	11/10/20	56	133
3	Historial de pedidos, bosquexos e pago de particulares	11/10/20	27/10/20	84	51
4	Pulido de funcionalidades	28/10/20	11/11/20	8	15
5	Redacción da memoria	11/11/20	7/5/21	-	96
Total					**505**

Táboa 5.1: Táboa de tempos do desenvolvemento da aplicación

5.3 Cálculo de costes

A partir das horas empregadas nas distintas iteracións que se póden observar na táboa 5.1 e dun custo medio aproximado de 30€/hora, calcúlase o coste do proxecto:

Custo total do proxecto = 505 horas * 30€/hora = **15150€**

Capítulo 6

Fundamentos Tecnolóxicos

Neste capítulo redactarase un resumo das tecnoloxías e ferramentas empregadas no proxecto.

6.1 Tecnoloxías empregadas no Backend

O backend da aplicación é unha API REST implementada con Java, utilizando o framework de Spring Boot e con base de datos MySQL entre outras ferramentas. A continuación descríbense a totalidade das mesmas.

6.1.1 Java

Java é unha das linguaxes máis utilizadas no mundo. É unha linguaxe de programación de propósito xeral, concurrente, con tipado forte e orientada a obxectos. A utilicei para o menu proxecto por ser a que me resulta máis familiar e porque xunto con Spring permíte un desenvolvemento rápido do backend dunha aplicación web.

6.1.2 MySQL

MySQL [8] é un xestor de base de datos relacional de código aberto e basado en SQL. Foi a miña opción principal porque, do mesmo xeito que Java, é o mais familiar a min, pero tamén por ser de código aberto e pola facilidade de atopar tutoriais ou axuda na rede se me eran precisos, xa que esta considerado o xestor de base de datos máis popular no mundo.

6.1.3 Maven

Maven [9] é unha ferramenta para a xestión e construción de proxectos Java. Proporciona unha forma de facilitar a xestión da construción dun proxecto software, a súa documentación, dependencias, as versións do software e a súa distribución.

Maven esta baseado no Project Object Model (POM) que é un arquivo eXtensible Markup Language (XML) que describe o proxecto e sobre o que se pode declarar: a versión do software, a localización do código fonte, as dependencias, plugin e distintos perfiles para a construcción do proxecto (Por exemplo, un perfil para test, desenvolvemento ou para o entorno de producción).

6.1.4 Eclipse

Eclipse [10] é o Integrated Development Enviroment (IDE) utilizado para o desenvolvemento do backend. Esta ferramenta ten a capacidade de soportar varias linguaxes de programación, pero é utilizada maioritariamente para o desenvolvemento en Java, é de código aberto e ten a capacidade de extenderse de xeito indefinido a través de plugins dispoñibles a través do propio IDE. Isto fai posible a integración de eclipse con outras ferramentas que se utilicen no desenvolvemento.

6.1.5 Spring Boot

Spring é un framework de código aberto que facilita o desenvolvemento de aplicacións Java. Este framework proporciona entre outros:

- Inxección de dependencias.
- Creación de controladores Web, tanto vistas como aplicacións REST.
- Acceso a datos con Java DataBase Connectivity (JDBC) ou un Object-Relational Mapping (ORM) como pode ser hibernate.
- Conten un framework de testing, con soporte para JUnit.

Spring Boot [11] é un dos proxectos máis importantes dentro do marco de Spring. A configuración inicial e a preparación das aplicacións para producción son tareas bastante tediosas e para as que Spring Boot simplifica o proceso ao máximo. Isto é posible grazas a dous motivos principais:

- Spring Boot pode compilar aplicacións como un arquivo .jar integrando un servidor de aplicacións en dito arquivo, o que fai moito máis sinxelo distribuílas.
- Proporciona unha serie de dependencias chamadas ”starters” que veñen con valores por defecto (Aínda que se poden cambiar ditos valores) e pretenden minimizar a necesidade de configuración á hora de desonvolver unha aplicación.

6.1.6 Paypal Payments API

A API REST de pagos de Paypal [12] permite facer pagos online de forma sinxela e segura. Na aplicación utilizáronse só 3 operacións de dita API, correspondentes ás funcions de ”crear un pedido”, que proporciona unha URL para pagar unha cantidade indicada de antemán pola miña aplicación, ”executar un pago” unha vez o usuario fai un pago para confirmalo e ”buscar pedido” simplemente para saber se o pago do cliente existe e ten o prezo que lle corresponde. Para isto foi necesario iniciar sesión co meu usuario de Paypal na súa páxina web, crear unha aplicación na mesma, e introducir as credenciais proporcionadas despois deste proceso na miña aplicación no backend.

6.2 Tecnoloxías empregadas no Frontend

O frontend da aplicación consiste nunha vista programada con Javascript cos frameworks de React e Redux entre outras tecnoloxías. A continuación detállanse todas ás empregadas.

6.2.1 Javascript

É unha linguaxe de programación interpretada con funcións de primeira clase (as funcións son tratadas como calqueira outra variable). É unha linguaxe baseada en prototipos, dinámica e que soporta tanto estilos de programación orientada a obxectos como imperativa ou declarativa (funcional).

```
1 const unhafuncion = function() {
2    console.log("meulog");
3 }
4 //
5 unhafuncion();
```

Listing 6.1: Exemplo de función asignada a unha constante

6.2.2 Visual Studio Code

Visual Studio Code [13] é o IDE utilizado no desenvolvemento do frontend. Este foi desenvolvido por Microsoft, é de código aberto, compatible con unha gran candidade de linguaxes de programación e conta con aspectos como depuración de código ou control de git integrado. Ademáis, dunha forma parecida a eclipse pódense engadir plugins para personalizalo ou engadir funcionalidades novas que poden ser moi útiles ou facilitar a integración con outra ferramentas.

6.2.3 Yarn

Yarn [14] é o xestor de dependencias utilizado para o frontend da aplicación web. Yarn, por defecto utiliza o rexistro de dependencias de NPM por defecto (outro xestor de dependencias moi similar) e as dependencias do frontend poden atoparse nun arquivo ”packaje.json” no que se poden engadir as dependencias a man ou engadilas mediante o comando ”yarn add”, que facilita moito o traballo.

6.2.4 React

React [15] é un framework que pretende axudar a crear interfaces de usuario interactivas de forma sinxela. Ten un enfoque declarativo e está baseado en distintos compoñentes encapsulados que teñen o seu propio estado. Estes compoñentes se combinan para formar interfaces de usuario complexas.

```
  class Timer extends React.Component {
  constructor(props) {
    super(props);
    this.state = { seconds: 0 };
  }

  tick() {
    this.setState(state => ({
      seconds: state.seconds + 1
    }));
  }

  componentDidMount() {
    this.interval = setInterval(() => this.tick(), 1000);
  }

  componentWillUnmount() {
    clearInterval(this.interval);
  }

  render() {
    return (
      <div>
        Segundos: {this.state.seconds}
      </div>
    );
  }
}

ReactDOM.render(
```

```
31    <Timer />,
32    document.getElementById('timer-example')
33  );
```

Listing 6.2: Compoñente de React con estado propio, obtido da páxina oficial de react

Como se pode observar no método render do compoñente anterior, utilizase JSX. JSX é unha extensión semántica a javascript que permite escribir expresiones en XML para describir o conxunto de elementos devoltos por un compoñente de React.

Por último, como aclaración importante para o funcionamento de React, creo necesario expoñer que todos estos compoñentes declarados en JSX non son renderizados directamente no Document Object Model (DOM) do navegador, xa que as súas actualizacións son moi costosas e se houbera demasiadas poderían entorpecer a experiencia do usuario final (cada vez que hai un cambio de estado hai que renderizar o compoñente de novo). Por isto, unha vez se cambia o estado os elementos declarados no render actualízanse nunha estrutura de datos en forma de árbore chamada virtual DOM (o cal é moito máis eficiente). Cando esto pasa, React calcula as diferenzas entre o estado anterior e o novo e logo aplica as diferencias ao DOM do navegador, co que se consegue que o estado se actualice de forma moito máis eficiente.

6.2.5 Redux

Redux [16]é unha librería que permite xestionar o estado da aplicación dunha forma centralizada e modular. Este non esta ligado a React e pode utilizarse con outros frameworks.

O obxectivo principal do uso de redux é extraer a maior parte do estado da aplicación dos compoñentes, prevendo desta forma albergar o dito estado nun compoñente único ou obligar a que o estado se tivera que pasar de compoñentes pais a fillos para poder obter o seu valor.

Redux conta cun obxecto ”store” que almacena o estado da aplicación, unha serie de accións declaradas polo programador que se chaman cada vez que se quere modificar o estado e un reductor que se encarga de producir un novo estado ante unha acción. Con todo isto, pódese xestionar o estado da aplicación e para poder acceder ao seu valor, calquer compoñente que o precise pode recibir una notificación cada vez que se produza un novo estado coa función ”store.subscribe” e ler dito estado con ”store.getState”

6.2.6 Material-UI

Material-UI [17] é un framework de código aberto que proporciona unha serie de compoñentes React que implementan Material Design.

Material Design [18] é un sistema de deseño de aplicacións web creado por Google coa intención de que estas sexan facilmente adaptables a distintos dispositivos. Os compoñentes de material design son bloques interactivos para crear a interfaz de usuario inspirados no

mundo físico, as súas texturas, en como reflicten a luz e crean sombras. Además, Material Design tamén pon atención no tema (Material Theming) das páxinas web e na concordancia que deben ter os compoñentes en relación aos seus cores e formas para manter unha interfaz atractiva para o usuario.

Todo isto permite un desenvolvemento rápido e sinxelo do frontend da miña aplicación web pero ao mesmo tempo facer unha interfaz de usuario atractiva para os seus usuarios.

6.3 Tecnoloxías complementarias

6.3.1 Git

Git [19] é unha das ferramentas de control de versións máis utilizados no mundo. É distribuído, no que cada usuario ten unha copia local do repositorio sobre a que fai os cambios aínda que neste caso non é de gran utilidade xa que o proxecto só é levado a cabo por unha persoa.

6.3.2 Redmine

Redmine [20] é unha aplicación web consistente nun sistema de control de tarefas. Esta ferramenta é de codigo aberto e permite engadir varias aplicacións con versións e tarefas en cada unha. Permite tamén modificar unha gran cantidade de información a cerca das tarefas dun proxecto e ver o seu Diagrama de Gantt. Todo isto, xunto con unha páxina de actividade, control de tempos, integración con Git e moitas outras fan a Redmine moi útil para levar a cabo a xestión dun proxecto software.

Capítulo 7

Desenvolvemento

Neste capítulo, en primeiro lugar farase un resumo da estrutura de ficheiros da aplicación, tanto do backend como do frontend. Logo, exporanse os aspectos máis relevantes do desenvolvemento da aplicación explicando o funcionamento da mesma, así como os seus compoñentes principais. Para isto farase un resumo das catro iteracións levadas a cabo durante o proxecto poñendo énfase nas fases de análise, deseño e implementación das mesmas.

7.1 Estrutura da aplicación

Como se trata dun proxecto maven, o backend ten un cartafol /src/main/java que conten o código fonte da aplicación. Dito cartafol esta dividido en dous principais paquetes:

- Model: que contén a capa de acceso a datos e polo tanto paquetes para as entidades, os DAOS, os servizos e as excepcións utilizadas por estes servizos
- Rest: que contén os controladores REST da aplicación, os Data Transfer Object (DTO) e un paquete "common" onde se gardan clases comúns como pode ser a configuración de Spring Security e o Json Web Token (JWT)

Ademais, no backend tamén existen os seguintes subdirectorios:

- /src/main/resources: contén ficheiros de internacionalización para as mensaxes de erro das excepcións do backend, o arquivo .yml para a configuración da aplicación e imaxes utilizadas pola mesma.
- /src/test/java: contén os test automatizados da aplicación.
- /src/test/resources: conten o arquivo .yml para a configuración da aplicación á hora de executar os test.

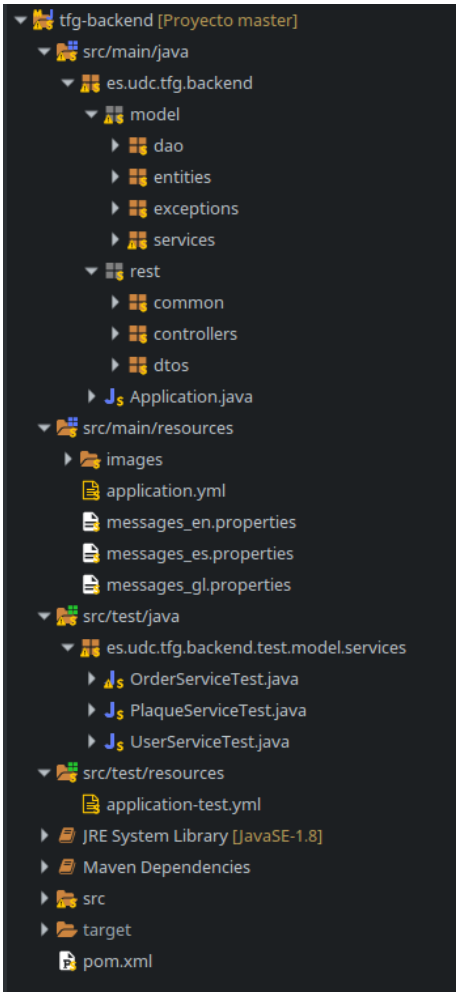

Figura 7.1: Distribución dos directorios do backend

Para o frontend dispoñemos dun cartafol /src que engloba todo o código fonte e que conten os seguintes elementos:

- /src/backend: encárgase da comunicación co frontend mediante a API fetch
- /src/i18n: contén os arquivos de internacionalización do frontend
- /src/modules: que contén os distintos módulos da aplicación. Ditos módulos conteñen sempre arquivos correspondentes a Redux para cambiar o estado da aplicación e un cartafol ”components” onde se gardan os compoñentes de React. Na aplicación hai en total un modulo ”App” encargado de formar a estrutura básica da aplicación (cabeceira, corpo, pé de paxina...), un módulo ”common” que contén os compoñentes comúns a toda a aplicación e un módulo para cada sección lóxica da nosa aplicación.

- index.js: compoñente de React no que se inicializa a store de Redux, establécese o locale e que se encarga de renderizar toda a vista.

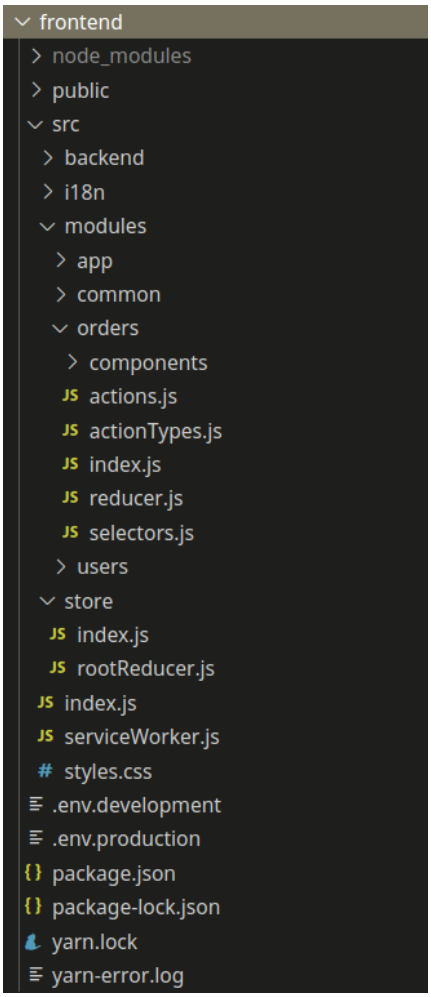

Figura 7.2: Distribución dos directorios do frontend

7.2 Iteración 1: Sección de administración

7.2.1 Análise

Na primeira iteración do desenvolvemento do proxecto o obxectivo era realizar os casos de uso correspondentes á aplicación de administración. Desta forma os empregados da empresa de arte funerario poden comezar a operar coa mesma engadindo clientes habituais e engadindo pedidos na aplicación de forma que sexan visibles para todos os empregados unha vez engadidos.

Os casos de uso que se abordaron nesta iteración son os seguintes:

- CU1 - Login
- CU3 - Pechar sesión
- CU10 - Engadir usuarios
- CU11 - Mostrar o historial de pedidos
- CU12 - Engadir pedido
- CU13 - Cambiar estado de pedido
- CU16 - Actualizar información de pedido
- CU17 - Eliminar pedido
- CU18 - Mostrar usuarios
- CU19 - Aplicar desconto
- CU20 - Modificar usuario
- CU21 - Eliminar usuario
- CU22 - Detalle de pedido

7.2.2 Deseño e implementación

Para comezar a iteración realizouse unha interface de usuario básica e xeral para toda a aplicación, de forma que a partir da mesma se puidera comezar o desenvolvemento dos casos de uso da aplicación de forma sinxela. Esta interface ten 3 compoñentes diferenciados, a cabeceira, o corpo e o pé de páxina da aplicación. A cabeceira esta composta polo título da aplicación e unha barra lateral que serve para á navegación entre diferentes vistas da aplicación e que está oculta ata pulsar un botón no caso dos dispositivos cunha pantalla de tamaño reducido. Por outra parte, o corpo é o encargado de renderizar o contido da aplicación e o pé de páxina consiste nun pequeno texto que da remate ao corpo. Por último, para mostrar máis en detalle a arquitectura do backend, amósanse imaxes que corresponden a un diagrama de clases das entidades da aplicación (figura 7.3) e outro diagrama das demais clases do backend, como son os controladores, servizos e DAOs (figura 7.4)

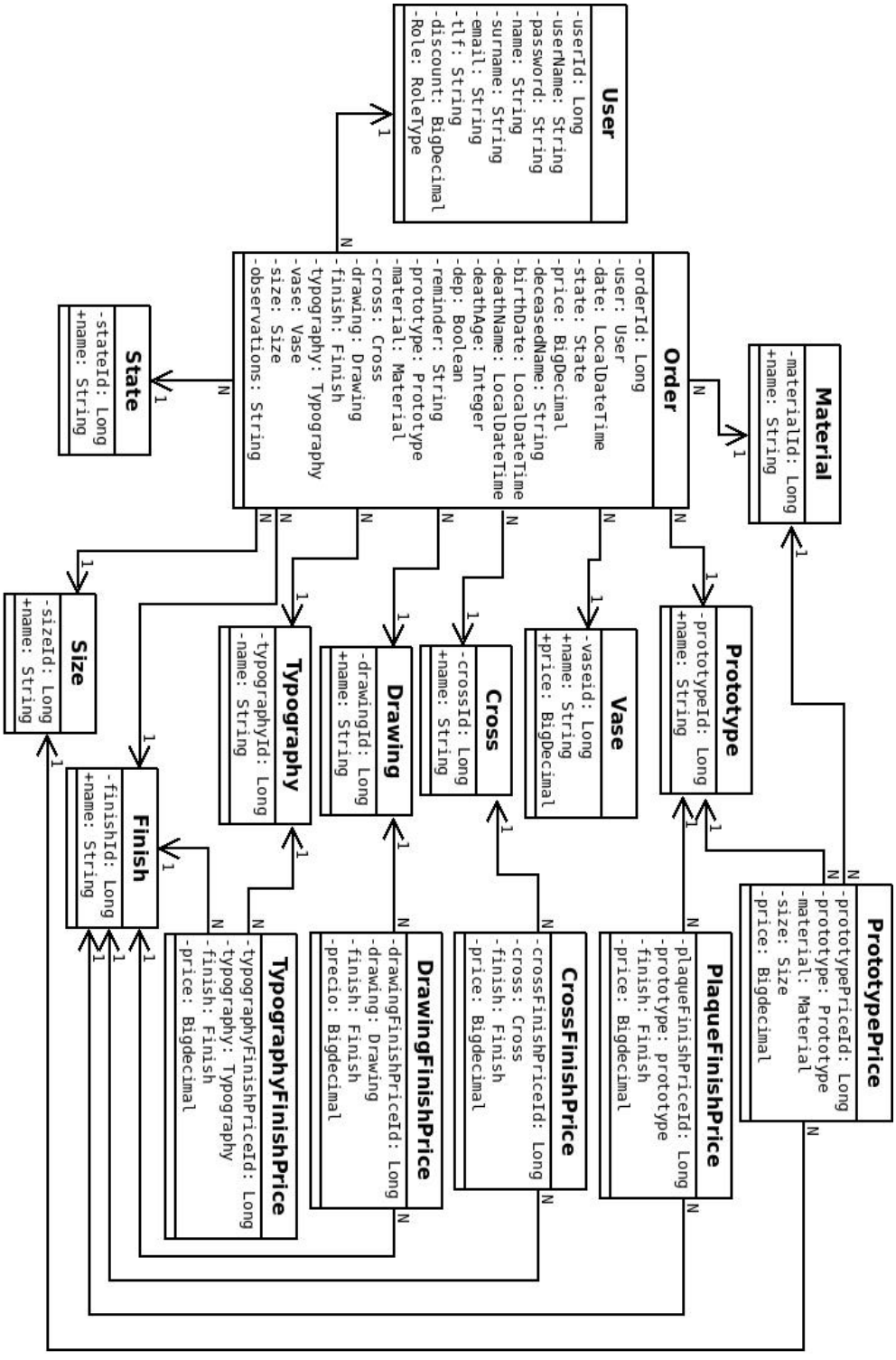

Figura 7.3: Diagrama de clases das entidades na iteración 1.

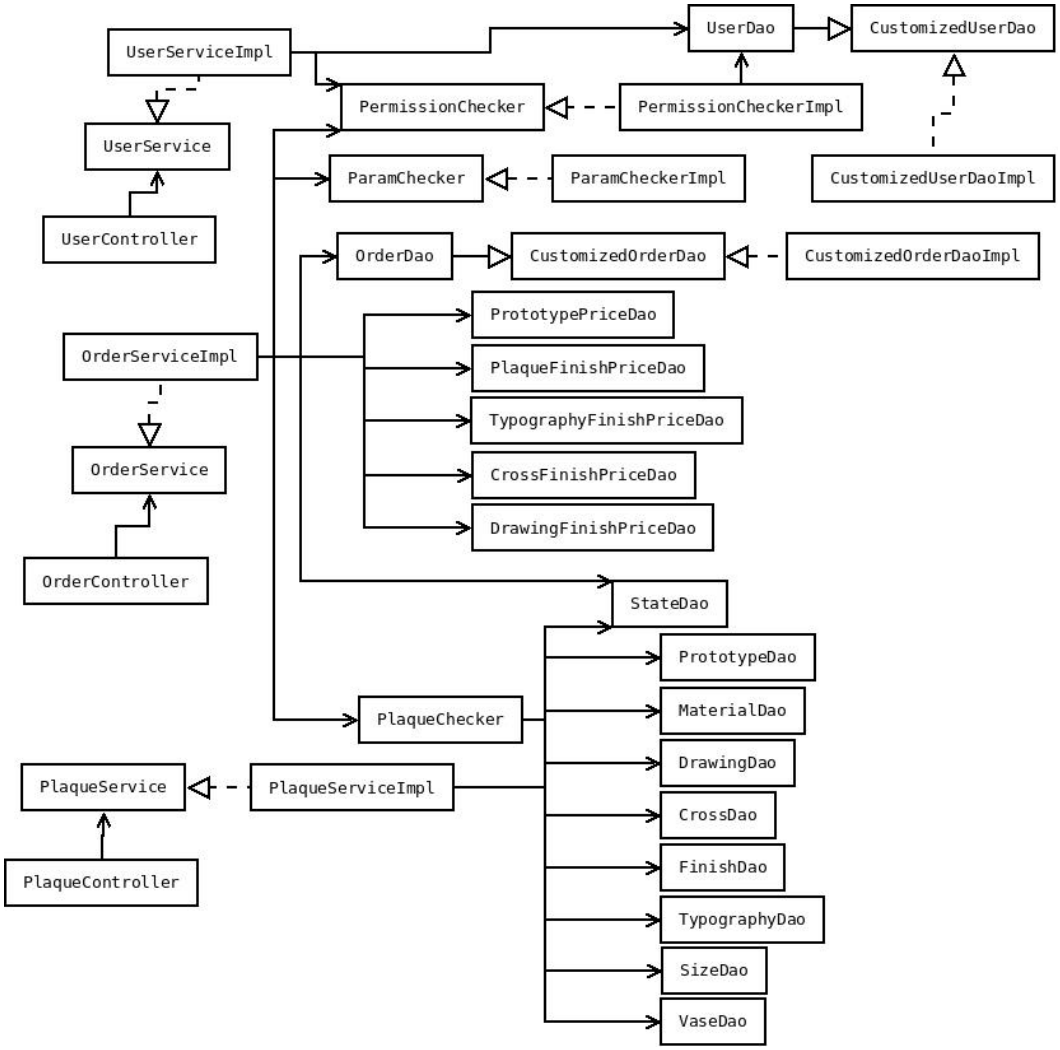

Figura 7.4: Diagrama de clases do backend na primeira iteración que mostra a arquitectura do mesmo.

CU1 - Login: Para implementar o login da aplicación web optouse polo estándar JWT [21]. Este estandar permite a creacións de tokens que manteñen o estado da autenticación dun usuario na aplicación, o que permite que o backend sexa totalmente stateless. Desta forma, o backend non necesita manter sesións en memoria co estado da autenticación, senón que se garda o estado ao completo no lado cliente, o cal se encarga de engadilo xunto a cada petición feita ao backend.

Os tokens JWT funcionan da seguinte forma:

- Primeiro, un usuario se autentica na aplicación web. Para isto, o backend encárgase de comprobar as súas credenciais.

- Se as credenciais son correctas crease un token JWT, que como mínimo inclúe un ID de usuario pero que neste caso componse dese ID e o rol do usuario.
- O backend envia como resposta o token ao usuario e este o garda no almacenamiento interno do navegador, de forma que se o frontend se recarga o usuario segue autenticado.
- A partir deste momento, cada vez que se fai unha petición ao backend envíase xunto a mesma o token JWT.
- Cando o backend recibe unha petición, este encárgase de comprobar a validez do token antes de permitir ao usuario realizar dita petición. Además, o backend tamén comproba que o usuario posexa o rol necesario para realizar a petición demandada.

Este último paso de control de acceso lévase a cabo mediante Spring Security. Grazas a esta ferramenta se comproba a validez do token JWT e se o usuario ten os roles necesarios para realizar una petición mediante un arquivo de configuración. A continuación móstrase un exemplo do arquivo de configuración de Spring Security (listing 7.1) e un diagrama de secuencia dos token JWT funcionando na aplicación (figura 7.5).

```
protected void configure(HttpSecurity http) throws Exception {
    http.cors().and().csrf().disable().sessionManagement()
        .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
        // A seguinte liña se encarga de validar
        // o token e ler o rol do usuario
        .and().addFilter(new JwtFilter(authenticationManager(),
        jwtGenerator)).authorizeRequests()
      // As liñas restantes comproban que o usuario teña
      // o rol necesario para a acción solicitada
    .antMatchers("/users/signUp").permitAll().
    .antMatchers("/users/login").permitAll()
    .antMatchers("/users/loginFromServiceToken").permitAll()
    .antMatchers("/plaque/*").permitAll()
    .antMatchers("/orders/buy").authenticated()
    .antMatchers("/orders/buysuccess").permitAll()
    .antMatchers("/orders/buycancel").permitAll()
    .antMatchers("/orders/userOrders").authenticated()
    .antMatchers("/orders/cancelOrder").authenticated()
    .antMatchers("/orders/createOrderPayment").authenticated()
    .antMatchers("/orders/orderPaymentSuccess").permitAll()
    .antMatchers("/orders/myOrderDetails/*").authenticated()
    .antMatchers("/orders/makeOrder").authenticated()
    .antMatchers("/orders/**").hasRole("ADMIN")
    .antMatchers("/users/addUser").hasRole("ADMIN")
    .antMatchers(HttpMethod.GET, "/users/**").hasRole("ADMIN")
```

```
26      .antMatchers("/users/*/updateUser").hasRole("ADMIN")
27      .antMatchers("/users/*/delete").hasRole("ADMIN")
28      .anyRequest().authenticated();
29  }
```

Listing 7.1: Exemplo de configuración de Spring Security

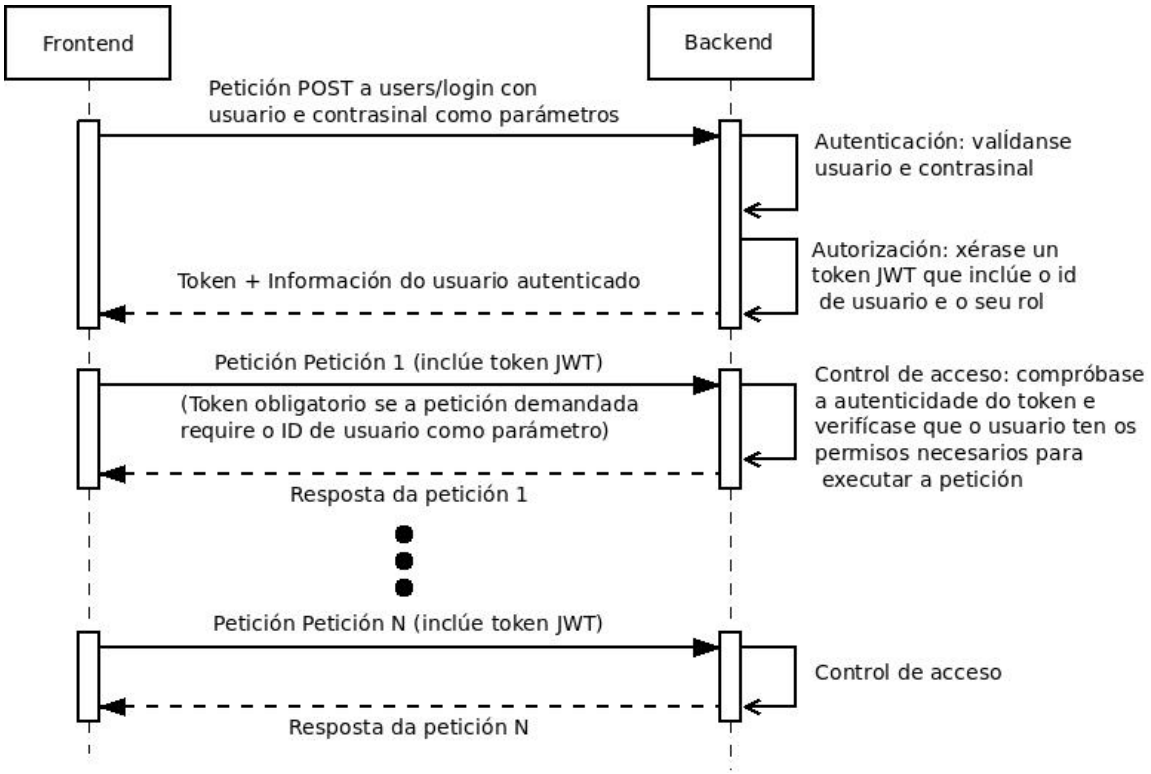

Figura 7.5: Diagrama de secuencia do uso dos token JWT na aplicación

No referente aos compoñentes utilizados para o caso do login, estes consisten soamente en dúas entradas e un botón que forman un formulario sinxelo co nome de usuario e un contrasinal. Ademáis, aínda non existe un acceso ao rexistro, xa que nesta iteración só estará dispoñible a sección de administración e os usuarios só poden ser engadidos mediante un usuario administrador ou directamente en base de datos. Para completar este caso de uso, unha vez autenticado un usuario, na cabeza da páxina aparece o seu nome de usuario na esquina dereita, coa posibilidade de realizar varias accións correspondentes a algúns dos casos de uso desta iteración. Na figura 7.6 móstrase o compoñente de login.

Figura 7.6: Vista do compoñente de login.

CU3 - Pechar sesión: Para este caso de uso engadiuse unha opción no botón dun usuario autenticado que pecha a sesión unha vez pulsado. A aparencia do botón pode observarse na figura 7.7. Para que o botón sexa funcional, soamente é necesario eliminar o token JWT do estado da aplicación no lado cliente e do Local Storage do navegador, xa que, como se mencionou anteriormente, o backend non se encarga de almacenar o estado da autenticación dos usuarios.

Figura 7.7: Botón de peche de sesión.

CU10 - Engadir usuarios: Para engadir novos usuarios na aplicación, implementouse unha operación POST na API REST do backend que recibe todos os datos dun usuario, incluido o seu rol. Tras verificar que os datos son correctos esta operación almacena un novo usuario co rol seleccionado.

No frontend engadiuse un novo botón para engadir usuarios na barra de navegación lateral. Esta leva a un compoñente ”AddUser” que consiste nun formulario composto por 7 ”TextField” de Material-UI, un despregable feito co compoñente ”Select” tamén de Material-UI e un botón para completar o engadido. Os TextField son inputs que aceptan texto como entrada por parte do usuario. Os TextField que se poden atopar neste formulario son o nome de usuario, o contrasinal, confirmación de contrasinal, nome, apelidos, email e telefono, mentres que o campo correspondente ao despregable é o do rol de usuario. Ademais, os campos de confirmación de contrasinal, email e teléfono teñen comprobacións adicionais para indicar ao usuario se a información introducida é correcta. A vista correspondente a este compoñente

”AddUser” pódese atopar na figura 7.8

Figura 7.8: Vista do compoñente AddUser.

Este compoñente AddUser é un controlled component, o que quere dicir que os datos do formulario se gardan no estado interno do propio compoñente (que é independente do resto da aplicación) ou no estado de redux, pero nunca directamente no DOM do navegador. Neste caso e nos demais formularios dispoñibles na aplicación web, todos os campos utilizan o hook de estado ”useState” para poder acceder ao estado de React sen a necesidade de escribir unha clase. Utilizando esta estratexia, declárase unha variable de estado coa que se pode acceder ao valor de cada campo e unha función para cada unha de ditas variables co propósito de actualizalas.

Esta estratexia é moi habitual en formularios como é o caso, xa que non é necesario acceder ao valor do seu estado dende outros compoñentes. Ademais, permite que no momento no que se actualiza un compoñente, se fagan comprobacións ou modificacións dos datos introducidos, como pode ser a validación dun email, proporcionando información útil ao usuario en caso de que cometa erros.

CU11 - Mostrar o historial de pedidos: Para mostrar o historial de pedidos optouse pola utilización dunha lista paxinada. Para isto, inicialmente implementouse unha operación na API REST do backend consistente na obtención dunha lista de pedidos a partir dos criterios de busca de nome de usuario, data (un rango de tempo especificando data de inicio e fin) e estado dun pedido. Esta operación recibe tamén o número de páxina para atopar os pedidos da páxina correspondente, a cal ten sempre un tamaño máximo de 10 pedidos.

No frontend engadiuse un botón no despregable de navegación que leva directamente ao historial de pedidos. Unha vez este botón se pulsa, faise unha petición da primeira páxina da lista de pedidos e dispárase unha acción de Redux para actualizar o estado conforme a nova lista de pedidos. Este botón pode atoparse na figura 7.9.

Figura 7.9: Vista do botón da para acceder á lista de pedidos.

Unha vez obtidos os pedidos e actualizado o estado, a lista toma forma grazas a un compoñente ”Table” de Material-UI que consiste nunha táboa de datos sinxela cunha cabeza ”TableHead” e un corpo ”TableBody”. A cabeza esta formada por nun compoñente ”TableRow” para dar forma á primeira lista da táboa, que a súa vez esta formada por un ”StyledTableCell” para cada campo dispoñible na lista de pedidos, facendo estos de título de cada columna da táboa (ID, nome de usuario, data, prezo...). O corpo consiste nun ”StyledTableRow” por cada pedido proporcionado ao facer unha busca, cada un de ditos pedidos ten á súa vez un ”StyledTableCell” para cada un dos seus campos, da mesma forma que na cabeza pero esta vez co seu valor correspondente en lugar do nome do campo, como é de esperar dunha táboa. Ademais, cabe destacar que ”StyledTableCell” e ”StyledTableRow” son compoñentes propios da aplicación, desenvoltos a partir dos existentes ”TableRow” e ”TableCell” de Material-UI para engadir estilo a ditos compoñentes e mostrar unha táboa acorde ao resto da aplicación. Na figura 7.10 podemos observar a vista e os distintos compoñentes da lista de pedidos mediante

unha captura do navegador, mentres que na figura 7.11 exponse un diagrama de compoñentes de dita vista.

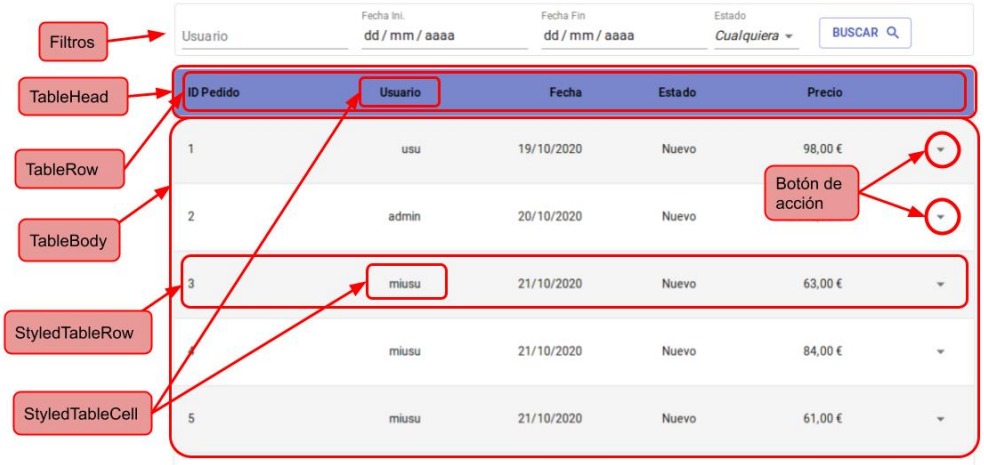

Figura 7.10: Vista do compoñente correspondente á lista de pedidos.

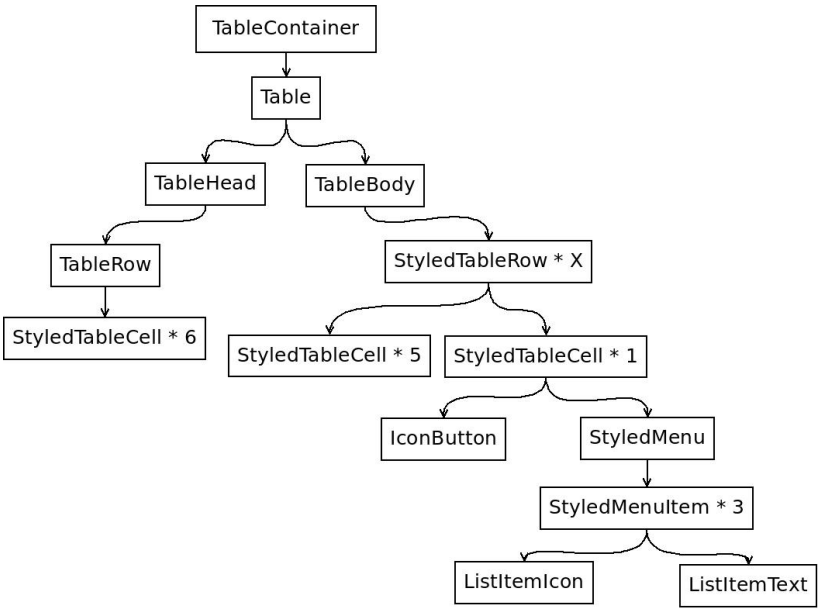

Figura 7.11: Diagrama de compoñentes da vista correspondente á lista de pedidos.

Este caso de uso complétase co engadido dos filtros de busca e o correspondente botón para buscar cos filtros na parte superior e os botóns correspondentes á paxinación na parte inferior da táboa. Estes botóns desencadean a mesma acción que a primeira vez que se abre o historial de pedidos, pero engaden novos parámetros correspondentes aos filtros ou o número da páxina solicitada para obter os pedidos axeitados á acción realizada polo usuario.

CU12, CU13, CU16 - Engadir pedido e actualizar pedido incluíndo o estado: Como consideracións iniciais da explicación para estes casos de uso, cabe destacar que se implementou a actualización do estado dun pedido de forma conxunta coa actualización dos demais campos, de forma que calquera tipo de actualización dun pedido queda centralizada nunha mesma vista. Ademais, optouse por reutilizar os compoñentes necesarios para engadir e actualizar, xa que o formulario necesario para ambas operacións é practicamente igual.

Nun primeiro momento implementáronse os botóns necesarios para acceder ao engadido e a actualización. Para o engadido bastou con un botón na barra lateral de navegación, mentres que para a actualización engadiuse un botón de acción para cada pedido na lista de pedidos. Ao ser presionado, dito botón de acción abre un pequeno despregable que permite acceder ás opcións dispoñibles para o pedido correspondente, entre as que se atopa o actualizado do mesmo. Ditos botóns pódense atopar na figura 7.12 Desta forma, se presionamos o botón de engadir pedido cárgase un compoñente "AddOrder" e se presionamos o botón de actualizado, cargase o compoñente "UpdateOrder", ambos encargados de renderizar o compoñente "OrderForm" (o compoñente xenérico para ambos casos de uso) que ten como parámetros un pedido e unha función a executar cando se complete o formulario. Así, o compoñente "AddOrder" renderiza "OrderForm" cun pedido nulo e unha función que engade un pedido, mentres que "UpdateOrder" chama ao backend para atopar o pedido co ID que se quere actualizar e o introduce como parámetro ao compoñente (isto ten como propósito mostrar os valores que teñen os campos do pedido a actualizar), asi como unha función para actualizar dito pedido.

Figura 7.12: Vista dos botóns para acceder ao engadido e á actualización.

Para a parte común a ambos casos de uso existe o compoñente "OrderForm". Este compoñente consiste nun formulario sinxelo que utiliza compoñentes de Material-UI e compoñentes propios da aplicación para os seus campos, que son: nome, formato das datas (para escoller como se mostra a data na placa de cemiterio), data de inicio, data de fin, idade, D.E.P, recordatorio, estado, observacións, prezo, pagado, prototipo, material, cruz, debuxo, acabado, tipografía, tamaño e floreiro. Para os campos consistentes nun campo de texto como o nome ou as observacións utilizouse o compoñente "Textfield" de Material-UI, para os campos booleanos

como D.E.P ou pagado utilizouse ”Checkbox”, para o prezo utilizouse un compoñente especial para moeda tamén propio de Material-UI chamado ”CurrencyTextField” e por último para os selectores optouse pola implementación dun novo compoñente ”GenericSelector” a partir do compoñente de Material-UI ”Selector”, que serve para facer un selector de elementos cunha cantidade de elementos indefinida dado como parámetro unha lista. Por último, este compoñente remata cun botón que se encarga de executar a función correspondente ao engadido ou actualizado dependendo da función que se introduciu como parámetro nos compoñentes ”AddOrder” e ”UpdateOrder”, o que desencadea unha chamada ao backend para persistir os datos. Nas figuras 7.13 e 7.14 pódense observar as vistas de engadir e actualizar pedido coas súas diferencias.

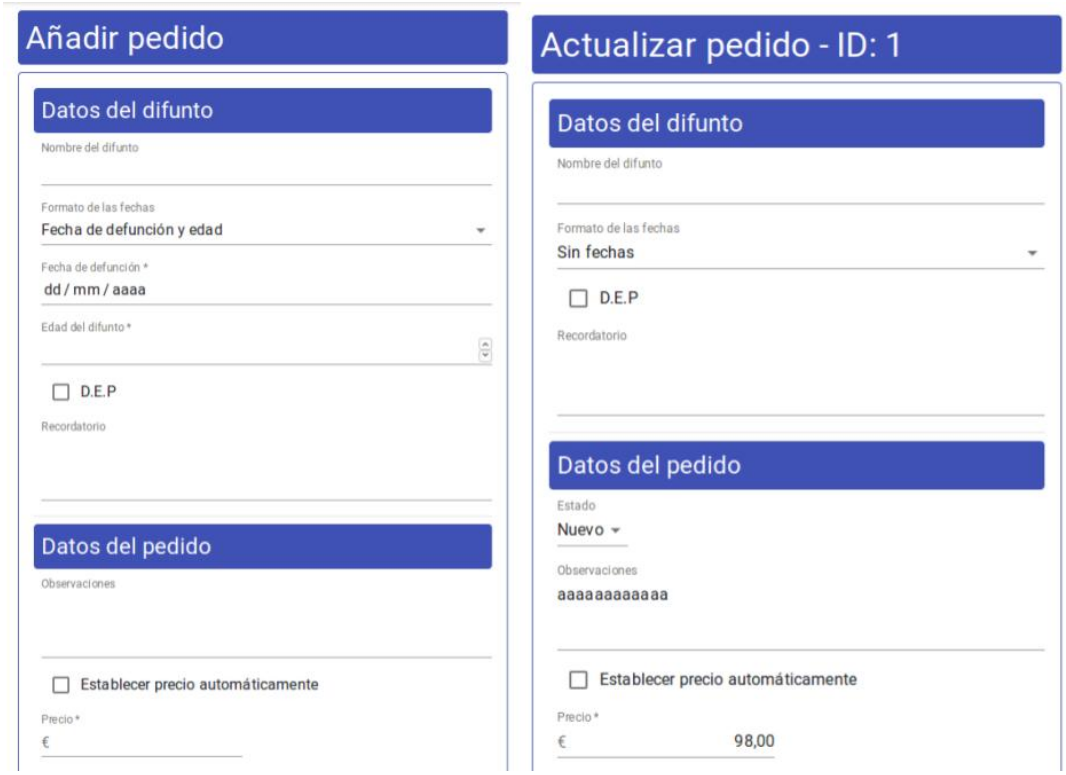

Figura 7.13: Vista da primeira parte do engadido e a actualización de pedidos.

Figura 7.14: Vista da segunda parte do engadido e a actualización de pedidos.

CU17 - Eliminar pedido: Para eliminar un pedido engadiuse un botón dentro do menú de acción da lista de pedidos. Desta forma, se un usuario pretende eliminar un pedido, buscará na lista de pedidos o que quere eliminar, premerá no botón de acción e escollerá o botón de eliminar pedido, que se pode atopar na figura 7.15. Isto abrira un diálogo, grazas a carga dun compoñente ”Dialog” de Material-UI que aporta seguridade no caso de que se prema o botón por erro e dá a opción de cancelar ou aceptar o eliminado do pedido. Se o usuario acepta eliminar o pedido, farase unha petición ao backend que eliminará o pedido na base de datos, o que fará que se recargue a lista de pedidos e se mostre unha mesaxe de confirmación. Pódese observar este diálogo de eliminado na figura 7.16

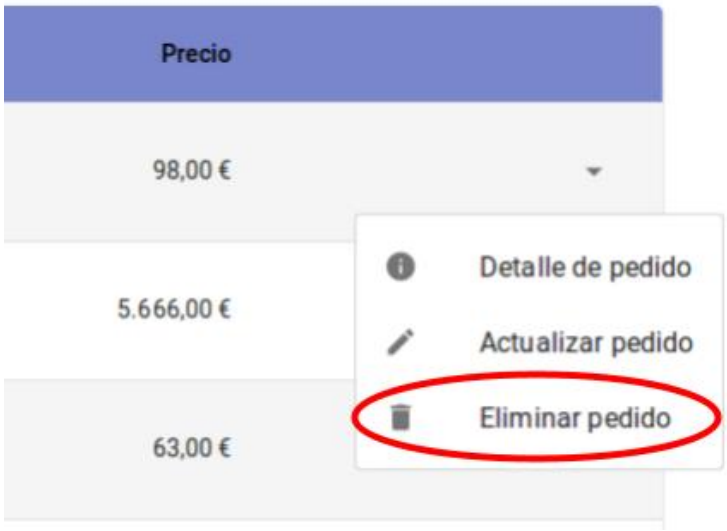

Figura 7.15: Vista do botón para acceder ao eliminado dun pedido.

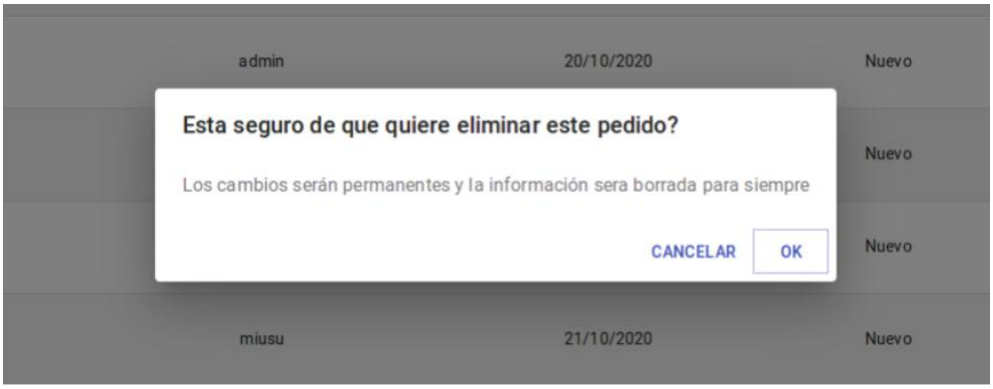

Figura 7.16: Vista do diálogo de eliminado de pedidos.

CU18, CU20, CU21 - Mostrar usuarios, modificar usuario e eliminar usuario: Estes tres casos de uso implementáronse dunha forma análoga aos casos de uso relativos aos pedidos. Desta forma, para mostrar os usuarios optouse por unha lista paxinada cun filtro de busca por nome e cun botón de acción en cada usuario que permite tanto modificar como eliminar usuarios. Os compoñentes utilizados para a lista son os mesmos que para a lista de pedidos (Table, TableHead, TableBody, StyledTableRow, StyledTableCell...) e a súa aparencia é practicamente igual. No caso do eliminado utilizase un diálogo igual ao de eliminar pedidos cambiando lixeiramente o texto de confirmación e no caso da actualización reutilizase o compoñente de engadido de usuarios da mesma forma que no engadido e actualización de pedidos, creando un compoñente común ambos chamado "UserForm". Nas figuras 7.17, 7.18 e 7.19 pódese ver a lista de usuarios, así como a modificación e a eliminación coas similitudes

con respecto aos casos de uso referentes aos pedidos.

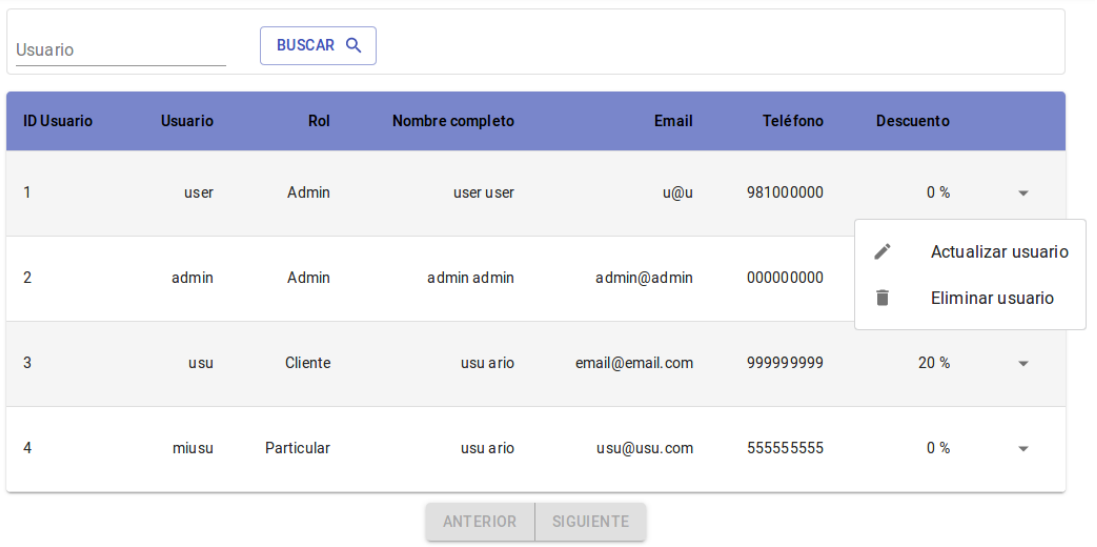

Figura 7.17: Vista da lista de usuarios dispoñibles na aplicación.

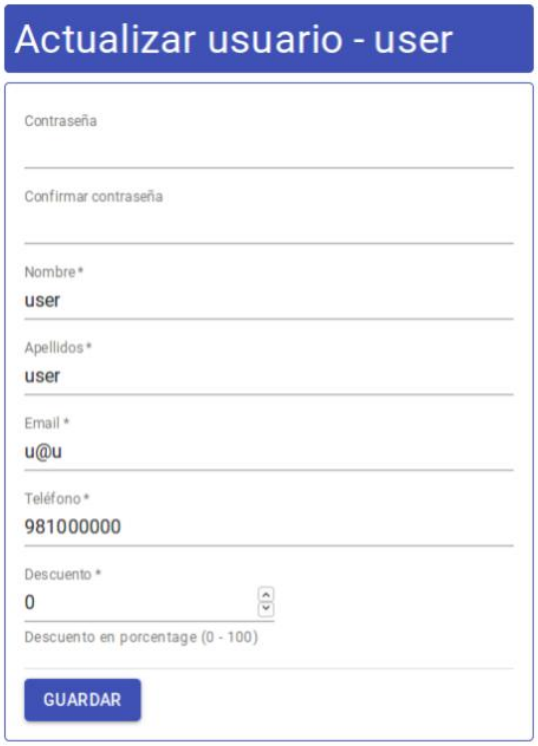

Figura 7.18: Vista do formulario de actualización de usuarios.

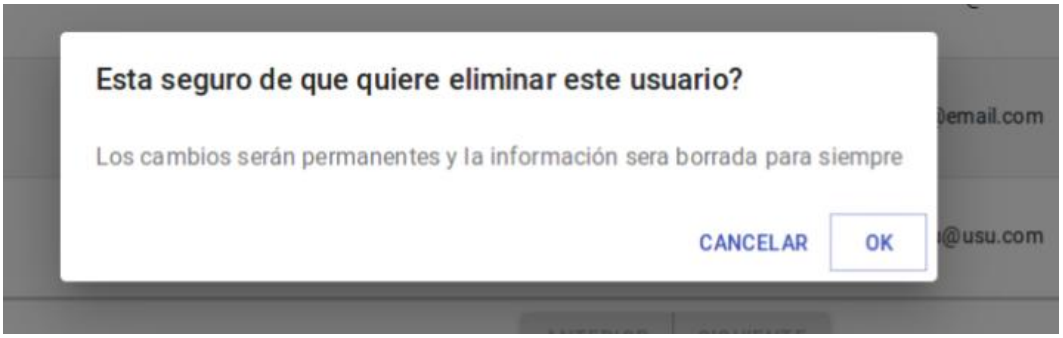

Figura 7.19: Vista do diálogo de eliminado de usuarios.

CU22 - Detalle de pedido Como na lista de pedidos non é posible mostrar todos os campos dos mesmos por falta de espazo, tomouse a decisión de incluir en dita lista só os mais importantes e engadir no botón de acción de cada pedido un apartado para mostrar dito pedido en detalle. Unha vez engadido o botón, implementouse un novo compoñente chamado ”OrderDetails” encargado de mostrar todos os campos dun pedido (obtidos mediante unha chamada ao backend, facendo unha busca do pedido polo seu ID). Estes campos móstranse mediante o compoñente ”List” de Material-UI, que se compón de outros compoñentes ”ListItem” e poden ter diferentes iconos e texto. Ademais, no caso do campo de prezo, utilízase o compoñente ”FormattedNumber” propio de React-Intl [22], que permite darlle formato de moeda de forma sinxela.

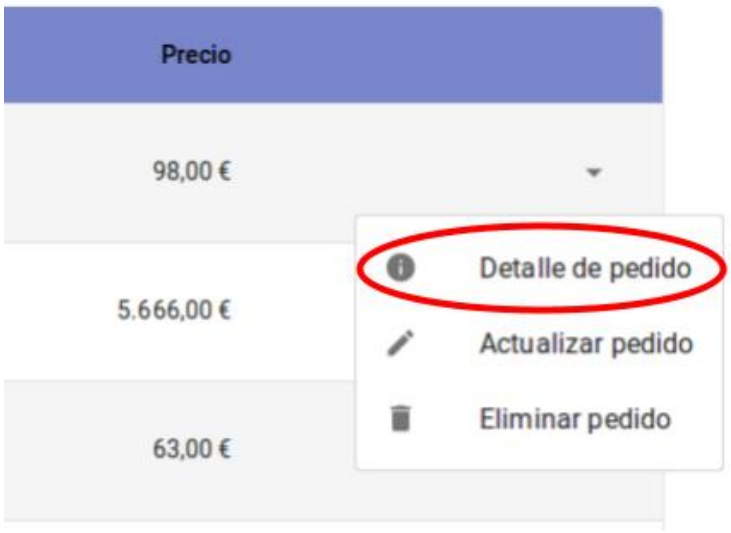

Figura 7.20: Vista do botón de acción co acceso ao detalle dun pedido.

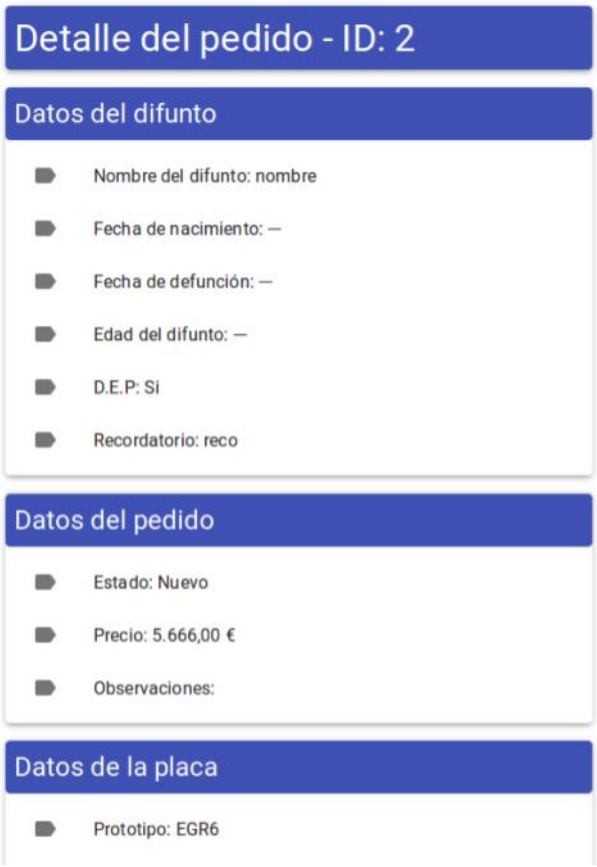

Figura 7.21: Vista do detalle dun pedido.

7.3 Iteración 2: Catálogo e realización de pedidos

7.3.1 Análise

O obxectivo da segunda iteración é o de dar a posibilidade de facer pedidos tanto a clientes como a particulares, que ata agora non tiñan acceso á aplicación web. Ademais, ao remate desta iteración tamén se pretende que os clientes podan pagar os seus pedidos. O motivo desta decisión foi que estes usuarios son os máis numerosos e que os usuarios particulares necesitan ver o bosquexo dun pedido para poder pagalo. Desta forma, a implementación do pagamento de pedidos feitos por particulares aumentaría moito a extensión desta iteración, polo que se implementará na seguinte.

Os casos de uso que se abordaron na segunda iteración son os seguintes:

- CU2 - Rexistro de usuarios particulares
- CU4 - Mostrar catálogo de placas

- CU5 - Facer pedido de placas de cemiterio

7.3.2 Deseño e implementación

CU2 - Rexistro de usuarios particulares Para o rexistro de particulares engadiuse un link no compoñente do login. Unha vez se fai click no link, accédese a un compoñente ”SignUp” que é un formulario con conta cos mesmos campos que o compoñente de engadir usuarios da sección de administradores a excepción do campo de rol. Unha vez se completa o formulario basta con facer unha chamada ao backend para persistir os datos e o usuario queda autenticado, entrando na páxina principal da aplicación que corresponde ao catálogo, que se explicará a continuación. A continuación móstrase o rexistro e o acceso ao mesmo na figura 7.22

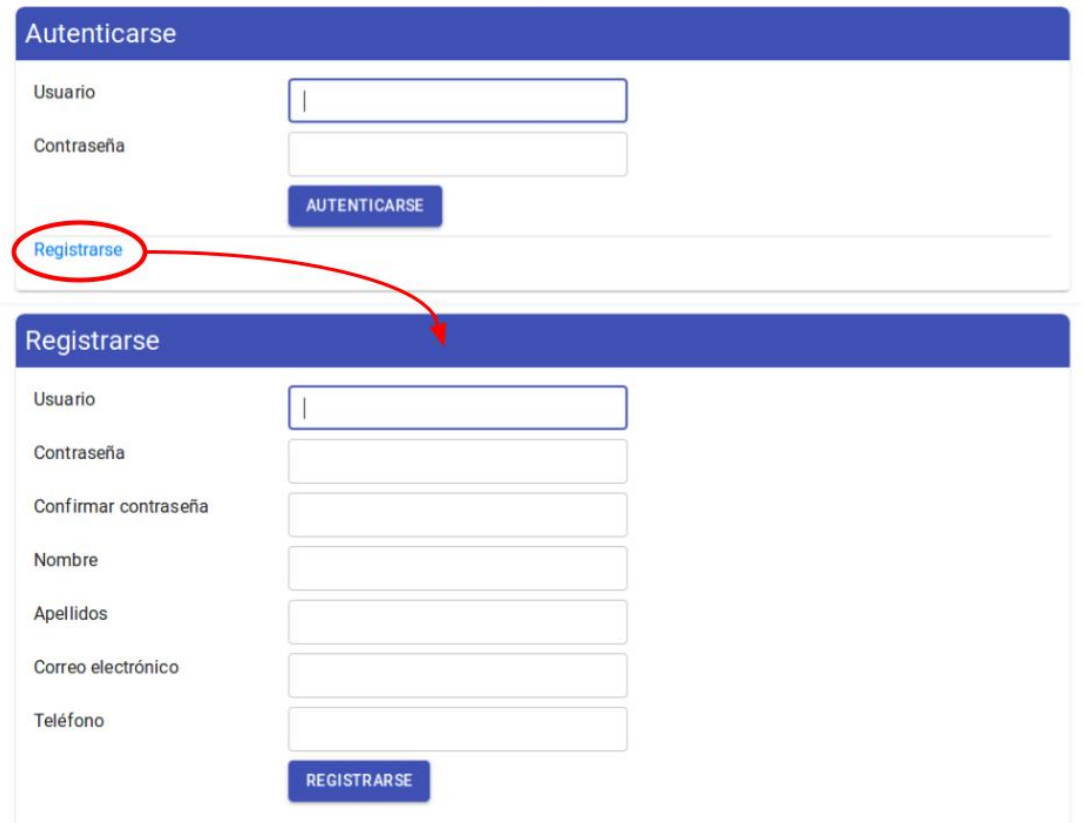

Figura 7.22: Vista do rexistro de usuarios praticulares, xunto co acceso ao mesmo.

CU4 - Mostrar catálogo de placas O catálogo de placas é a páxina principal da aplicación, xa que é a que se mostra cando calquera usuario sen autenticar accede á mesma por primeira vez. Para realizar este caso de uso foi necesario engadir fotos dos prototipos de placas de cemiterio na base de datos, xa que ata este momento non tiñan este campo, de forma que cando se entra na páxina e se fai unha petición para obter os prototipos teñan incluída dita

imaxe.

No referente ao frontend, o catálogo toma forma grazas a un compoñente ”GridList” que consiste nunha cuadricula de elementos que serán as fotografías dos prototipos de placas de cemiterio. Estes elementos son compoñentes ”GridListTile”, que se utilizan para mostrar unha imaxe e un ”GridListTileBar”, que a súa vez úsase para mostrar un texto a modo de pé de fotografía e un botón se é necesario. Todo isto resúmese nunha cuadricula de imaxes con unha barra gris na parte inferior a cal contén o nome do prototipo da placa e un botón para facer un pedido da mesma placa. Na figura 7.23 amósase unha imaxe do catálogo.

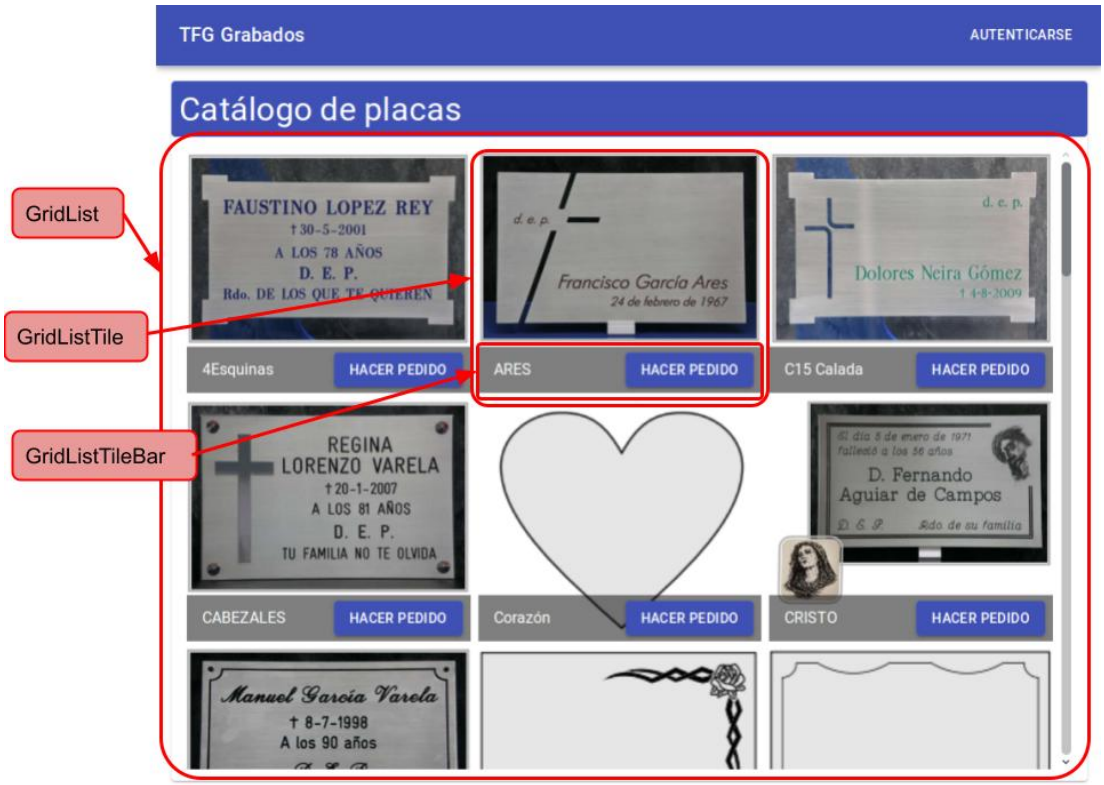

Figura 7.23: Catálogo de placas cos seus compoñentes.

CU5 - Facer pedido de placas de cemiterio Para facer un pedido dunha placa de cemiterio, nun inicio o usuario cliente ou particular mirará o catálogo e escollerá unha placa pulsando no seu botón correspondente para facer un pedido, o usuario necesitará autenticarse se non o está xa e comezará a personalizar o seu pedido. Para isto, se lle mostra a foto e o nome da placa que van a personalizar xunto cos selectores dos parámetros máis básicos da placa, que son tamaño, material e acabado e os prezos da placa escollida, que son o tamaño+material, xa que o prezo dos mesmos calcúlase de forma conxunta, o acabado e o prezo total. A vista

correspondente atópase na figura 7.24. Ademais, para cargar o compoñente anterior, ”BasicPlaqueInfoSelector” utilizase outro compoñente ”BasicPlaqueInfoSelectorInit” que se encarga da carga dos materiais, tamaños e acabados dispoñibles para a placa da que esta a facerse o pedido, así como os prezos dos mesmos.

Figura 7.24: Selector dos parámetros básicos dun pedido (material, tamaño e acabado).

Unha vez escollidos os parámetros básicos da placa lévase ao usuario á seguinte pantalla, que consiste principalmente nun compoñente ”Stepper” de Material-UI. Este compoñente guía ao usuario por unha serie de pasos que debe seguir para completar, neste caso, o pedido e mostra un contido diferente dependendo do paso no que se atope. Os pasos que se necesitan para completar dito pedido son 6: Información do defunto, Seleccionar cruz, Seleccionar Debuxo, Seleccionar Floreiro, Observacións e Confirmar pedido.

Figura 7.25: Compoñente stepper utilizado para facer pedidos.

O paso de información do defunto consiste nun pequeno formulario parecido á parte correspondente na sección de administración e que ten os seguintes campos: nome do defunto, formato das datas, data de nacemento, data de defunción, idade de morte, D.E.P, recordatorio e tipografía, que esta acompañada do seu prezo e se engade ao prezo da placa base (figura 7.26). Para os campos de selección de cruz, debuxo e floreiro fíxose un compoñente común parecido ao que se pode atopar na pantalla do catálogo de placas (figura 7.27). Este compoñente común pretende mostrar imaxes e un botón en cada unha para seleccionar a desexada a partir dun listado de obxectos e unha función a executar cando se presione o botón dalgunha das imaxes. Unha vez seleccionada unha imaxe neste compoñente aparecerá o prezo do elemento seleccionado e sumarase ao prezo da placa básica. O paso de observacións consiste unicamente nun campo de texto grande e é o derradeiro paso que fará cambios no pedido (figura 7.28). Por último, a confirmación do pedido consiste nunha lista dos parámetros escollidos do pedido e o prezo final do mesmo, ademais dun botón para proceder ao pagamento (figura 7.29).

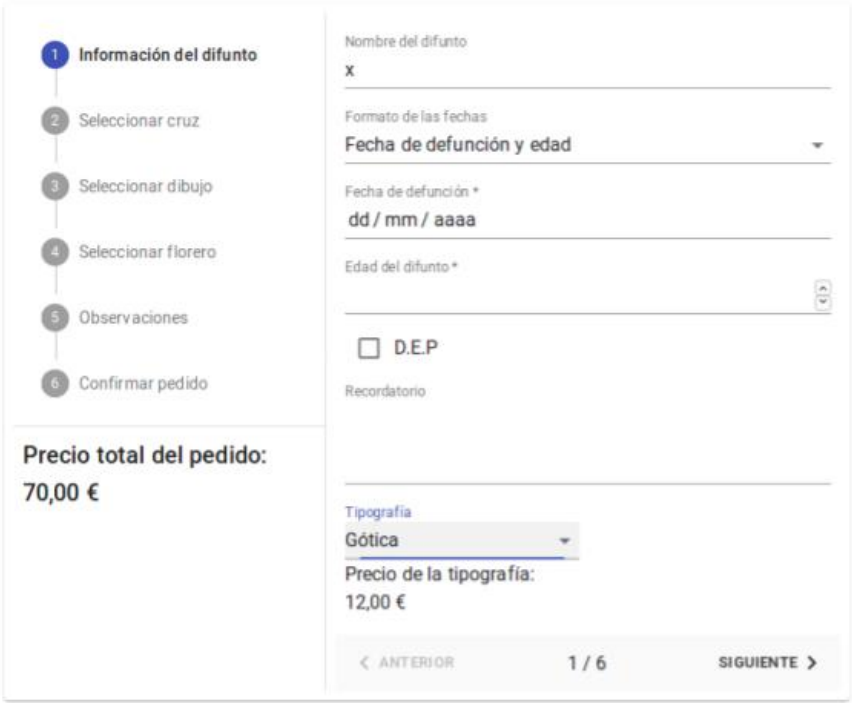

Figura 7.26: Selección da información do defunto ao facer un pedido.

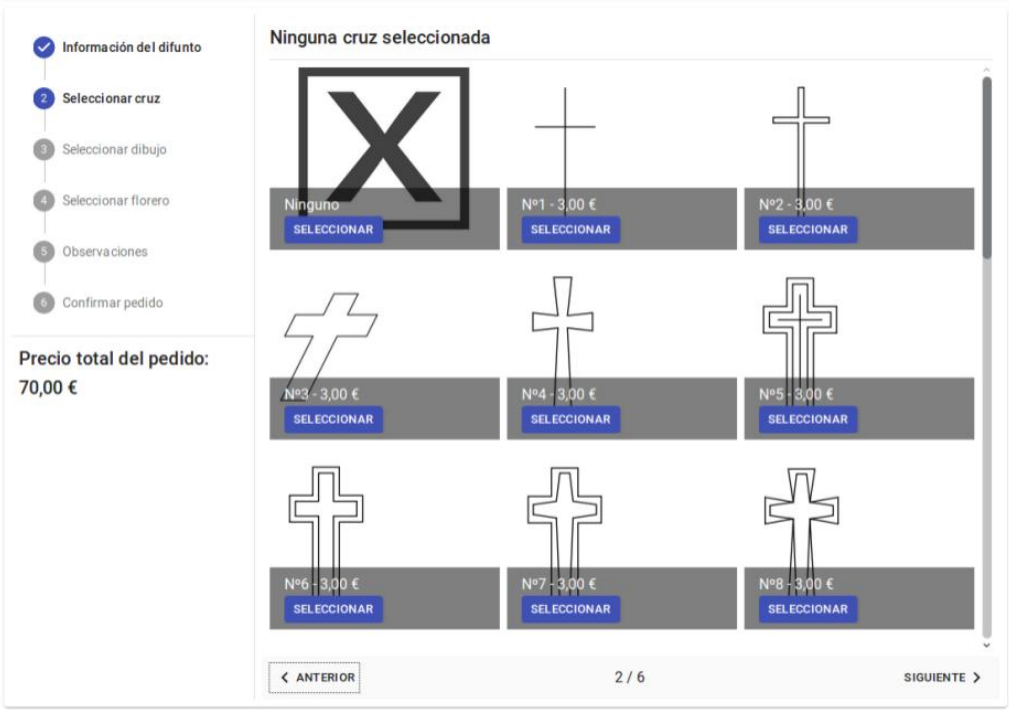

Figura 7.27: Exemplo dun dos selectores de compoñentes da placa ao facer un pedido.

Figura 7.28: Observacións feitas polo usuario ao facer un pedido.

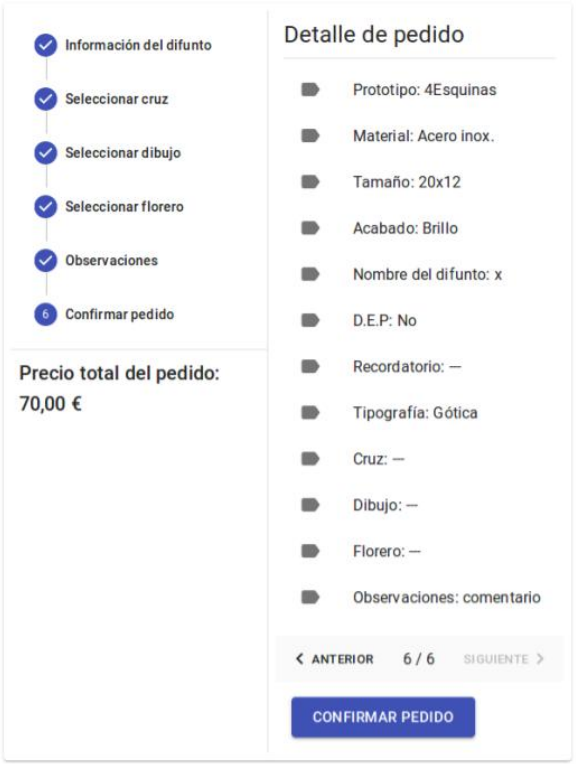

Figura 7.29: Detalle do pedido e confirmación do mesmo.

Unha vez se confirma o pedido, se o usuario é un particular farase unha petición ao backend para crear un novo pedido non pagado e se non hai ningún erro móstrase unha pantalla de confirma que o pedido foi engadido. Se o usuario que confirma o pedido é un cliente, procédese a pagar o pedido mediante Paypal. Para isto séguense os seguintes pasos:

- Faise unha chamada a un novo método da API REST do backend que se encarga de crear un novo pagamento mediante a API de pagamentos de Paypal logo de comprobar que todos os parámetros introducidos son correctos. A petición a dita API necesita unha URL de acerto e outra de erro para redirixir ao usuario unha vez termina o pagamento.

- A API de Paypal, ante esta petición dá como resposta unha URL que corresponde a un pagamento na web de Paypal.

- O frontend se encarga de utilizar a URL do pagamento para redirixir ao usuario á web de Paypal unha vez se actualiza o estado de Redux coa mesma. Unha vez na web de Paypal o usuario pode escoller o método de pagamento que máis lle interese, ver á información correspondente ao pedido e completar ou cancelar o mesmo. Isto pode verse na figura 7.30. Se o pedido se completa, este quédase pendente de confirmación.

- Ao completar ou cancelar o pedido na web de paypal rediríxese ao usuario á URL de acerto ou erro respectivamente, que no noso caso son dous URL do frontend, ambas correspondentes a unha mensaxe sinxela para proporcionar información ao usuario.

- Se o pedido se completa e o usuario se atopa na vista do mensaxe de acerto (figura 7.31), é dicir, de ”pedido completado”, o frontend fai unha nova petición ao backend que se encarga de crear un novo pedido pagado cos parámetros introducidos polo cliente. Para isto execútase o pagamento de paypal, polo que se completa a transacción monetaria, valídanse os campos do pedido engadido para evitar datos inválidos e engádese finalmente o pedido a nome do usuario correspondente.

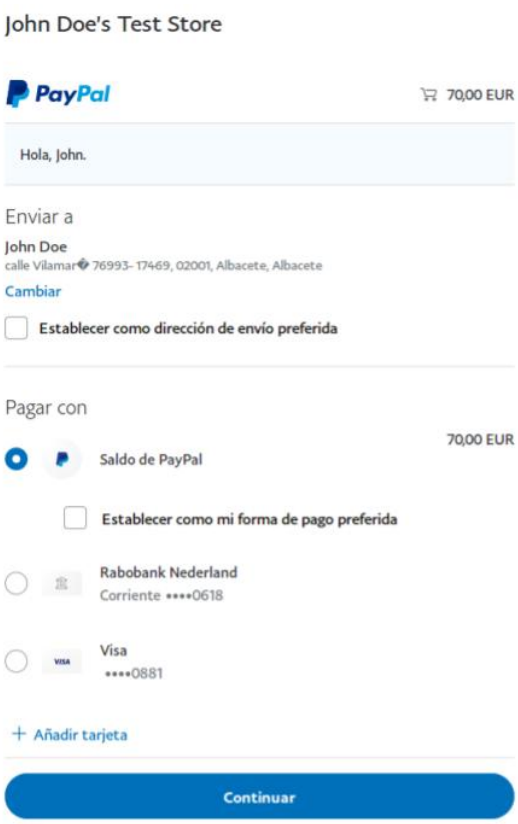

Figura 7.30: Opcións de pagamento na web de Paypal logo de autenticarse cun usuario de test.

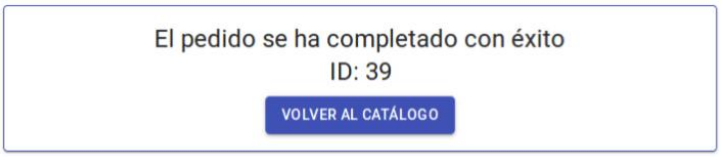

Figura 7.31: Mesaxe de pedido completado logo de pagar un pedido.

A continuación, amósase un diagrama de secuencia na figura 7.32 para ofrecer unha visión máis clara da explicación anterior.

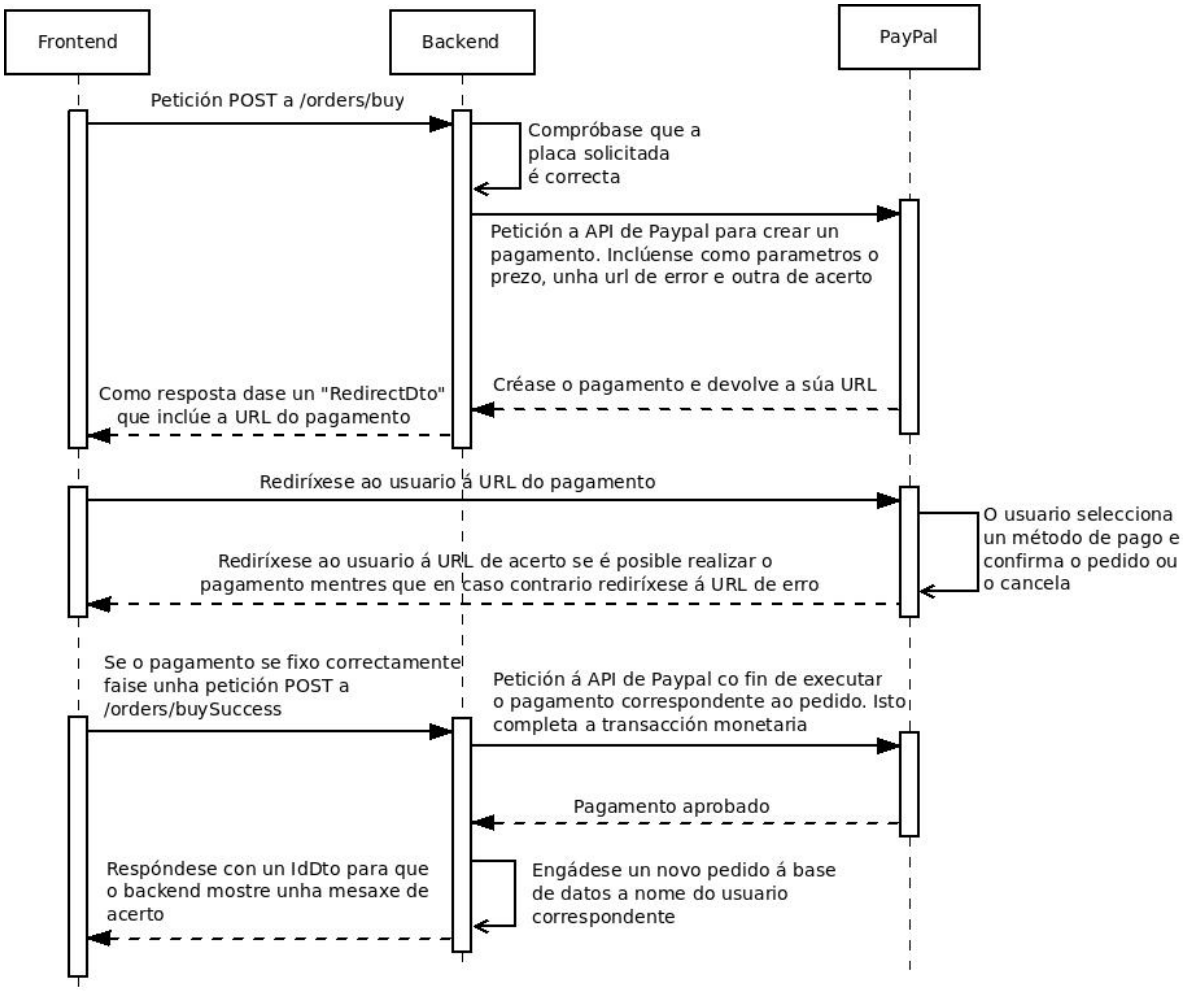

Figura 7.32: Diagrama de secuencia da interacción do frontend, backend e PayPal á hora de pagar un pedido

7.4 Iteración 3: Historial de pedidos con bosquexos e pagamento de pedidos dos particulares

7.4.1 Análise

O obxectivo desta terceira iteración foi permitir aos usuarios acceder a un historial dos seus pedidos e permitir pagar dende este historial os pedidos que non estiveran pagados aos usuarios particulares, que aínda non podían pagar unha vez feito o pedido. Para isto tamén foi necesario dar a posibilidade de engadir un bosquexo aos pedidos para que os particulares poidan ver unha versión simplificada da súa placa antes de pagar para evitar desconformidades.

Os casos de uso realizados nesta iteración son:

- CU6 - Cancelar pedido de placa de cemiterio
- CU7 - Ver bosquexo de pedido
- CU8 - Confirmar pedido e pagar
- CU9 - Mostrar pedidos de usuario
- CU14 - Engadir bosquexo
- CU15 - Actualizar bosquexo

7.4.2 Deseño e implementación

CU9 - Mostrar pedidos de usuario Para mostrar os pedidos dun usuario optouse por unha lista paxinada moi parecida á utilizada na sección de administración no listado de pedidos e usuarios. Desta forma, utilizouse un compoñente "Table" da mesma forma que na sección de administración, facendo unha lista cos campos de: ID, bosquexo, data, estado, prezo e pagado. Ademais, tamén se engadiu un botón de acción que permite ver o detalle do pedido, reutilizando o compoñente de detalle da sección de administración. Na figura 7.33 pódese observar a lista de pedidos de un usuario particular e un usuario cliente.

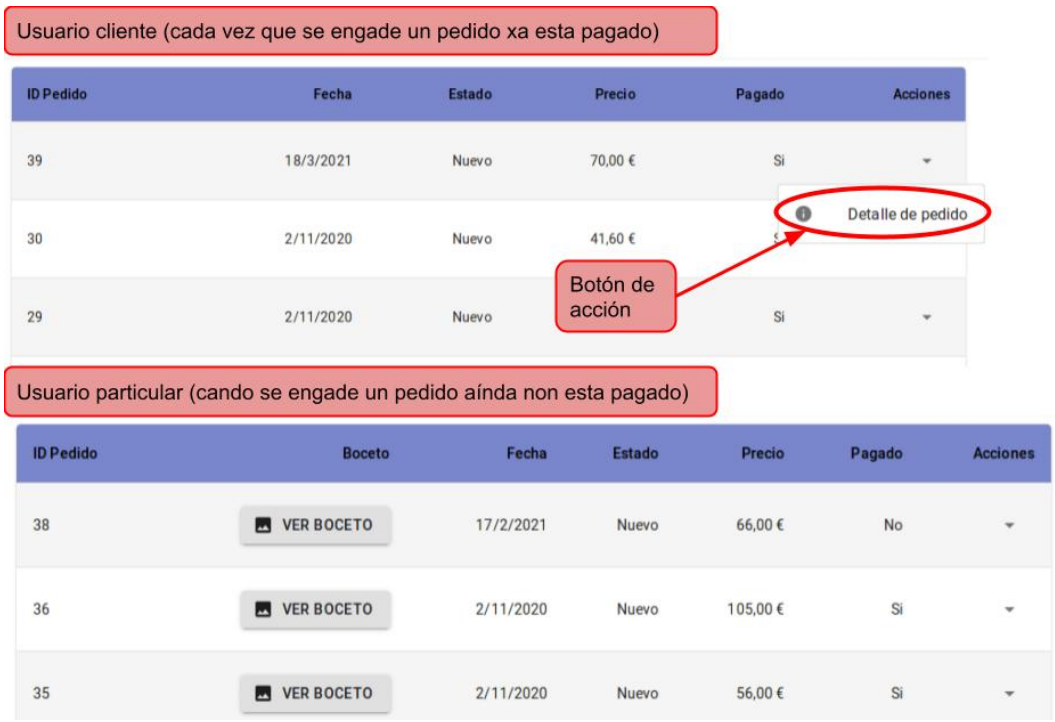

Figura 7.33: Pedidos de usuario cliente e particular coas súas diferencias.

Para obter a lista de pedidos, desenvolveuse un método na interface REST do backend que devolve os pedidos feitos polo usuario que fai a petición ordenados pola data na que se fan os pedidos. Esta lista obtense mediante un compoñente "MyOrders" que fai a petición REST e encárgase de mostrar o compoñente da lista de pedidos de usuario.

CU6 - Cancelar pedido de placa de cemiterio Para poder cancelar un pedido, engadiuse no listado de pedidos dun usuario, no botón de acción propio de cada pedido, un botón novo para cancelar pedido, que se pode ver na figura 7.34. Este botón só aparece dispoñible se o pedido que se quere cancelar non esta pagado. Unha vez presionado dito botón, faise unha petición ao backend para borrar o pedido e se procede con éxito, aparece unha mensaxe de confirmación. Neste caso, optouse por non pedir a confirmación do usuario para cancelar un pedido, xa que ao non estar pagado aínda, se o usuario o borra accidentalmente bastaría con facer un novo pedido para emendar o erro. Na figura 7.34 pódese observar o botón de cancelación de pedido na lista de pedidos dun usuario.

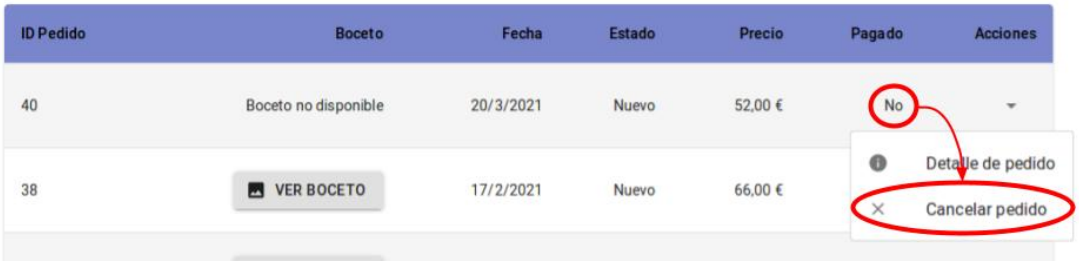

Figura 7.34: Botón de cancelanción de pedido na lista de pedidos dun usuario.

CU14, CU15 - Engadir e actualizar bosquexo de pedido Estes dous casos de uso fixéronse como un único botón na sección de administración, xa que actualizar o bosquexo dun pedido é actualizalo de nada a unha foto calquera. Este botón esta dispoñible no listado de pedidos no botón de acción de cada un dos pedidos feitos por particulares e pódese ver na figura 7.35. Unha vez pulsado o botón aparece un compoñente de tipo "Dialog" da mesma forma que no eliminado dun pedido ou usuario, pero que contén neste caso un input que acepta imaxes ".PNG" e ".jpeg". Unha vez introducida unha imaxe actívase un botón para aceptar a actualización do bosquexo e que en caso de ser pulsado fai unha petición REST ao backend para actualizar o pedido e persistir os datos. Se o backend non devolve ningún erro, aparece unha mensaxe de confirmación. Este dialogo de actualización de bosquexo pódese observar na figura 7.36.

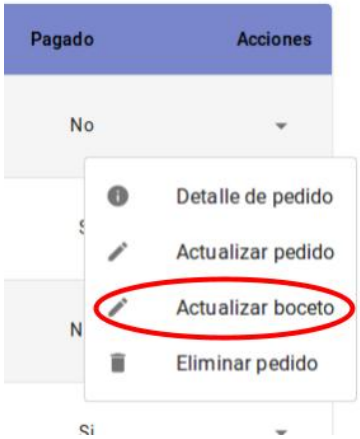

Figura 7.35: Botón de acceso a actualización do bosquexo dun pedido

Figura 7.36: Actualización do bosquexo dun pedido

CU7 - Ver bosquexo de pedido Para ver o bosquexo dun pedido, engadiuse na lista de pedidos, tanto da sección de administración como de usuario, un novo campo correspondente ao bosquexo. Se esta dispoñible aparecerá un botón na columna correspondente a ese campo de bosquexo que abrira un dialogo que se pode observar na figura 7.37, cunha cruz para cerralo na parte superior e a imaxe do bosquexo na parte inferior.

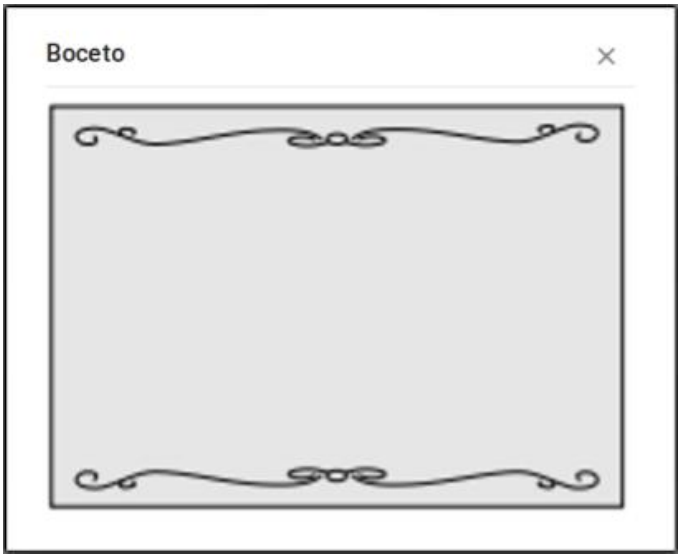

Figura 7.37: Vista do bosquexo dun pedido

CU8 - Confirmar pedido e pagar Para os pedidos non pagados da lista dos pedidos dun usuario, aparecerá no botón de acción da mesma forma que na cancelación un novo botón para pagar o pedido, sempre que este teña un bosquexo engadido, como se observa na figura 7.38. Isto débese a que un particular sempre ten que poder ver o bosquexo dun pedido antes de poder pagalo. Unha vez pulsado o botón procédese ao pagamento mediante Paypal. Para isto rediríxese ao usuario á autenticación na web de Paypal e séguense os mesmos pasos que se realizan cando un cliente fai un pedido.

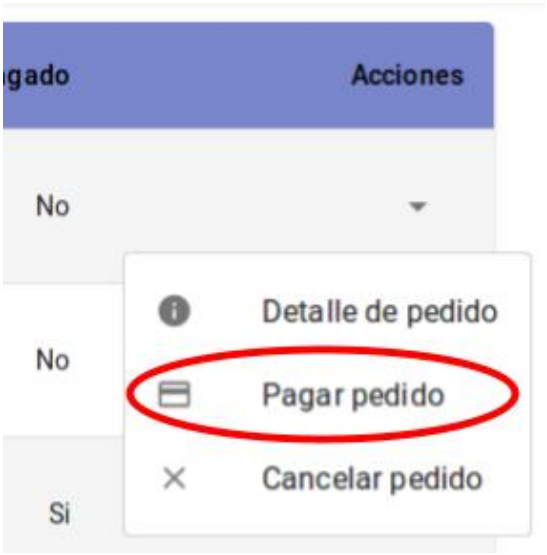

Figura 7.38: Botón para pagar pedido dispoñible na lista de pedidos dun usuario

7.5 Iteración 4: Solución de erros na aplicación

Esta foi a última iteración que se fixo para a aplicación web, pero o seu cometido non era implementar casos de uso novos, senón que pretendía pulir as funcionalidades realizadas ao longo das tres iteracións anteriores e mellorar distintos aspectos menores da aplicación.

Algunha das melloras mais importantes realizadas na aplicación foron:

- A busca de usuarios e pedidos por nome de usuario non recoñecía máis que o nome exacto, polo que se fixo máis flexible a consulta á base de datos.
- Engadiuse o campo de nome de usuario ao compoñente de detalle de pedido onde faltaba.
- Engadíronse mensaxes de internacionalización que faltaban no frontend.
- As listas de pedido non estaban ordenadas pola data, polo que se modificou o DAO de pedidos.
- Na selección de materiais, tamaño e acabado das placas era posible que quedara o prezo da anterior placa encargada cando se accedía ao compoñente, polo que se fixeron cambios á xestión do estado de Redux do compoñente.

Capítulo 8

Conclusións

8.1 Conclusións

Durante este traballo cumpriuse o obxectivo principal de desenvolver unha aplicación web SPA moderna e coas funcionalidades marcadas como obxectivo durante a concepción do proxecto. Ademais, o desenvolvemento da aplicación web foi moi útil para a miña formación, xa que aprendín moito sobre javascript e distintas formas de programar o frontend.

Material-UI foi unha axuda importante á hora de realizar o proxecto, xa que os seus compoñentes ofrecían unha forma sinxela de programar a vista da aplicación, ao proveer unha gran cantidade de compoñentes moi útiles a partir dos cales foi posible programar os meus propios. Esta ferramenta gustoume realmente polo sinxelo que era implementar os compoñentes e porque unha vez implementados podíanse entender nunha rápida ollada. Ademais estes compoñentes aínda estando feitos con relativamente pouco esforzo con relación á súa complexidade eran igualmente agradables á vista e deixan un acabado profesional.

O uso de Paypal tamén foi moi instructivo, xa que esta aplicación me permitiu ter unha primeira experiencia cunha pasarela de pago real. Esta ferramenta permitiu implementar o pago dunha forma rápida e segura, a pesar de que ten como contrapartida que Paypal cobra unha parte de cada pedido que se paga na aplicación. Aínda que a API de Paypal en xeral me pareceu moi interesante, un aspecto negativo a ter en conta é que a documentación da mesma é demasiado extensa e está mal organizada. Isto orixinou que a duración do proxecto na segunda iteración fose mais longa do estimado incialmente.

8.2 Traballo futuro

O traballo realizado podería estenderse coas seguintes liñas de traballo futuro:

- Para acoutar o alcance do traballo de fin de grao, a aplicación de administración non permite engadir novos compoñentes dunha placa e estes necesariamente teñen que ser

engadidos manualmente na base de datos, polo que a funcionalidade máis útil das que se pode engadir sería o engadido de novos compoñentes de placa dende a sección de administración.

- O engadido de novos filtros de busca para as listas de usuarios e pedidos sería de gran utilidade. Por exemplo, poderíase engadir un filtro que estivera activado por defecto e que consistiría en ocultar os pedidos xa completados na aplicación, xa que estes son de menos interese para os administradores.
- Poderíase engadir na mesma páxina que o catálogo unha sección coas placas máis vendidas na aplicación para mostrar aos clientes a información mais relevante. Se esta funcionalidade se desenvolve un pouco tamén se poderían mostrar distintas estatísticas das ventas aos usuarios administradores.

Apéndices

Relación de Acrónimos

API Application Programming Interface. 2, 23

DAO Data Access Object. 61

DOM Document Object Model. 27, 38

DTO Data Transfer Object. 29

IDE Integrated Development Enviroment. 24

JDBC Java DataBase Connectivity. 24

JWT Json Web Token. 29, 34

ORM Object-Relational Mapping. 24

POM Project Object Model. 24

REST Representational State Transfer. 2, 23, 29

SPA Single Page Application. 2

XML eXtensible Markup Language. 24

Glosario

backend Parte dunha aplicación encargada da xestión dos datos e a lóxica de negocio.. 2, 29

control de versións Sitema que rexistra os cambios realizacios nun conxunto de arquivos ao longo do tempo, permitindo a recuperación das distintas versións.. 28

Diagrama de Gantt Diagrama que ten como obxectivo expoñer o tempo de dedicacion previsto dun conxunto de tarefas ao longo dun tempo total determinado.. 28

framework Conxunto de ferramentas e módulos que permiten ser reutilizados en varios proxectos, permiten axilizar os procesos de desenvolvemento software e reducir código repetido.. 23

frontend Parte dunha aplicación encargada que se encarga da interacción directa co usuario, así como mostrar unha intarface gráfica.. 2, 29

hook característica propia de React que permite utilizar o estado e outras características sen utilizar unha clase. 38

metodoloxía iterativa Metodoloxía de desenvolvemento software creado en resposta ás debilidades do modelo tradicional en cascada que se basea nun conxunto de tareas agrupadas en pequenas etapas repetitivas chamadas iteracións.. 9, 19

metodoloxía áxil Metodoloxías de desenvolvemento software iterativas que permiten unha adaptación rápida aos cambios nos requisitos dun proxecto.. 9

mockup modelo do deseño visual dun sistema. 12

NPM Sistema de xestión de paquetes por defecto para Node.js e un entorno de execución para Javascript. 26

plugin elemento software que engade unha característica adicional a un sistema.. 24

SCRUM Metodoloxía áxil e iterativa que se basea na utilización de equipos de desenvolvemento autoorganizados que dividen o seu traballo en ciclos cortos chamados sprints. Estos sprints rematan nun producto potencialmente entregable e permiten unha resposta rápida aos cambios nos requisitos dun proxecto.. 9

Bibliografía

[1] "Páxina web de lasplacas.com." [En liña]. Dispoñible en: https://www.lasplacas.com/

[2] "Páxina web de rotumax." [En liña]. Dispoñible en: https://www.rotumax.es/

[3] "Páxina web do taller de arte renaud gravure." [En liña]. Dispoñible en: https://www.plaque-funeraire.fr/

[4] "Páxina web de gravinco." [En liña]. Dispoñible en: http://www.grabinco.com/

[5] "Páxina web de trofeos cadenas." [En liña]. Dispoñible en: https://trofeoscadenas.com/

[6] "Páxina web de lápidas vimianzo." [En liña]. Dispoñible en: http://www.lapidasvimianzo.com/

[7] "Páxina web de placasgravadas.es." [En liña]. Dispoñible en: https://placasgrabadas.es/placas_sepulturas.html

[8] "Páxina web de mysql." [En liña]. Dispoñible en: https://www.mysql.com/

[9] "Páxina web de maven." [En liña]. Dispoñible en: https://maven.apache.org/

[10] "Páxina web de eclipse." [En liña]. Dispoñible en: https://www.eclipse.org/

[11] "Páxina web de spring boot." [En liña]. Dispoñible en: https://spring.io/projects/spring-boot

[12] "Api de pagamentos de paypal." [En liña]. Dispoñible en: https://developer.paypal.com/docs/api/payments/v2/

[13] "Páxina web de visual studio code." [En liña]. Dispoñible en: https://code.visualstudio.com/

[14] "Páxina web de yarn." [En liña]. Dispoñible en: https://yarnpkg.com/

[15] "Páxina web de react." [En liña]. Dispoñible en: https://es.reactjs.org/

[16] "Páxina web de redux." [En liña]. Dispoñible en: https://es.redux.js.org/

[17] "Páxina web de material-ui." [En liña]. Dispoñible en: https://material-ui.com/es/

[18] "Páxina web de material design." [En liña]. Dispoñible en: https://material.io/

[19] "Páxina web de git." [En liña]. Dispoñible en: https://git-scm.com/

[20] "Páxina web de redmine." [En liña]. Dispoñible en: https://www.redmine.org/

[21] "Páxina web dos jwt." [En liña]. Dispoñible en: https://jwt.io/

[22] "Documentación de react-intl." [En liña]. Dispoñible en: https://formatjs.io/docs/react-intl/#the-react-intl-module