Facultade de Informática
UNIVERSIDADE DA CORUÑA

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

Aplicación web para a organización persoal de tarefas baseada en taboleiros kanban

Estudante: Diego Dorrego Díaz
Dirección: Fernando Bellas Permuy

A Coruña, febreiro de 2022.

Á miña familia e amigos

Agradecementos

En primeiro lugar agraceder á miña familia polo apoio brindado, e en especial a meus pais por axudar a facer posible que eu chegara ata aquí.

Aos meus amigos, polos bos momentos pasados e os que están por vir.

A Fernando, polos consellos e ideas recibidos ao longo deste proxecto.

Resumo

O obxectivo deste proxecto consiste no deseño e implementación dunha aplicación web inspirada nunha necesidade persoal de organizar e medir a miña dedicación ás tarefas que realizo. A aplicación debe permitir crear proxectos, con cadansúas tarefas, planificar estas tarefas usando taboleiros kanban, así como almacenar rexistros do tempo dedicado ás diferentes tarefas e proxectos, que se empregarán para xenerar informes de utilidade para o usuario. A diferencia de unha aplicación típica de xestión de tarefas, os taboleiros Kanban non representan un proxecto, representan un intervalo temporal, e poden conter tarefas de varios proxectos.

O backend da aplicación, o cal contén a lóxica de negocio desta, consiste nunha API REST implementada usando tecnoloxías Java, xunto co framework Spring Boot, persistindo os datos necesarios en MySQL.

O fontend da aplicación consiste nunha aplicación web SPA implementada empregando JavaScript, apoiándose en librerías como React, Redux ou Material-UI.

Abstract

The goal of this project is to design and implement a web application inspired by a personal need to organize and measure my dedication to the tasks I do. The application must allow to create projects, each with its own tasks, plan these tasks using kanban boards, as well as store the time dedicated to the tasks and projects, which will be used to generate useful reports to the user. Unlike a typical task management app, Kanban boards do not represent a project, they represent a time interval, and they can contain tasks from multiple projects.

The application backend, which contains its business logic, consists in a REST API implemented using Java tecnologies along with the Spring Boot framework, persisting needed data on MySQL.

The application's frontend consists of a SPA web application implemented using JavaScript, relying on libraries like React, Redux or Material-UI.

Palabras chave:

- Aplicación web SPA
- Java
- Spring Boot
- JavaScript
- React
- Kanban
- Tarefa

Keywords:

- SPA web application
- Java
- Spring Boot
- JavaScript
- React
- Kanban
- Task

Índice Xeral

Índice de Figuras

Índice de Táboas

Capítulo 1

Introdución

1.1 Contexto

Durante a miña etapa cursando o grao, e sobre todo nos dous últimos anos, nos que as asignaturas demandaban unha maior dedicación autónoma pola miña parte, comezou a complicárseme a labor de organizar, planificar e ter certo seguemento sobre as diferentes tarefas que debía facer.

En concreto, en cada cuadrimestre cursaba cinco asignaturas, nas que debía dedicar tempo tanto á comprensión dos conceptos teóricos como á realización das prácticas propostas. Polo tanto, o normal en cada semana era ter que dedicarlle tempo tanto a realizar avances nas prácticas como na parte teórica. Para isto, o máis cómodo para min sería ter a posibilidade de planificar nun mesmo lugar as tarefas das diferentes asignaturas e prácticas. Precisaba dunha ferramenta que me ofrecera maior flexibilidade á hora de definir o intervalo temporal no cal planificaba as tarefas, así como a posibilidade de telas organizadas en proxectos que foran independentes dun taboleiro.

En consecuencia, comecei a probar diferentes aplicacións de xestión de tarefas que, xunto coa axenda que xa usaba, me proporcionaran unha solución mellor adaptada ás miñas necesidades. Puiden ver que no contexto de aplicacións de xestión de tarefas existe un gran número delas, con unha boa variedade de métodos de xestión. Non obstante, despois de probar diferentes aplicacións ao longo de varios meses, observei que as ferramentas actuais non modelan ben a miña situación, xa que a cada aplicación que probara lle faltaba algo que outra si que tiña e eu necesitaba, e que para obter o que eu quería debía usar dúas aplicacións, duplicando tamén certos esforzos á hora de planificar.

Foi entonces cando me surxiu a idea de desenrolar unha aplicación de xestión de tarefas que xuntara as funcionalidades que máis me interesaban das diferentes aplicacións que probara, podendo así centralizar a planificación nunha única aplicación.

1.2 Obxectivos

Tendo en conta a situación anterior, o obxectivo deste proxecto é crear unha aplicación web de xestión de tarefas que permita que unha persoa se organice adecuadamente e saiba exactamente en que emprega o seu tempo.

Para isto, un usuario rexistrado poderá crear proxectos, que estarán compostos por tarefas, as cales poderán ter unha checklist, unha data límite ou unha porcentaxe de completitude por exemplo.

Por outra parte, poderá crear taboleiros Kanban que, a diferencia dunha aplicación tradicional de xestión de tarefas, corresponderán a un rango temporal, tendo unha data de inicio e unha de fin. Isto significa que un taboleiro estará a modelar as tarefas nas que o usuario ten pensado traballar nun intervalo de tempo (1 día, 3 días, 1 semana, etc). Polo tanto, nesta aplicación os taboleiros son independentes dos proxectos, polo que poderán conter tarefas dos diferentes proxectos que hai. Esta é outra característica diferenciadora con respecto ás aplicacións tradicionais de xestión de tarefas, nas que un taboleiro está ligado a un proxecto, e en consecuencia só pode ter tarefas do mesmo. Por outra parte, ao igual que nas aplicacións tradicionais de xestión de tarefas, os taboleiros estarán compostos por columnas, nas cales, como se mencionou, se poderán dispoñer as tarefas dos diferentes proxectos, correspondendo cada columna ao estado no que se atopan as tarefas contidas na mesma.

Para levar o control do seu tempo, poderase empregar un cronómetro, que deberá estar correndo mentres se lle dedica tempo a unha tarefa, así como se poderán engadir rexistros de tempo manualmente ás tarefas. Con esta información a aplicación xerará unha serie de gráficas que lle permitirán ao usuario observar dunha forma visual, por exemplo, a que dedica o seu tempo.

Esta aplicación de xestión de tarefas permite modelar a situación exposta no apartado anterior. Na figura 1.1 ilústrase cómo se poderían planificar prácticas de diferentes materias do primeiro cuadrimestre de cuarto deste grao en taboleiros Kanban temporais, coas tarefas xestionadas en proxectos externos a estes.

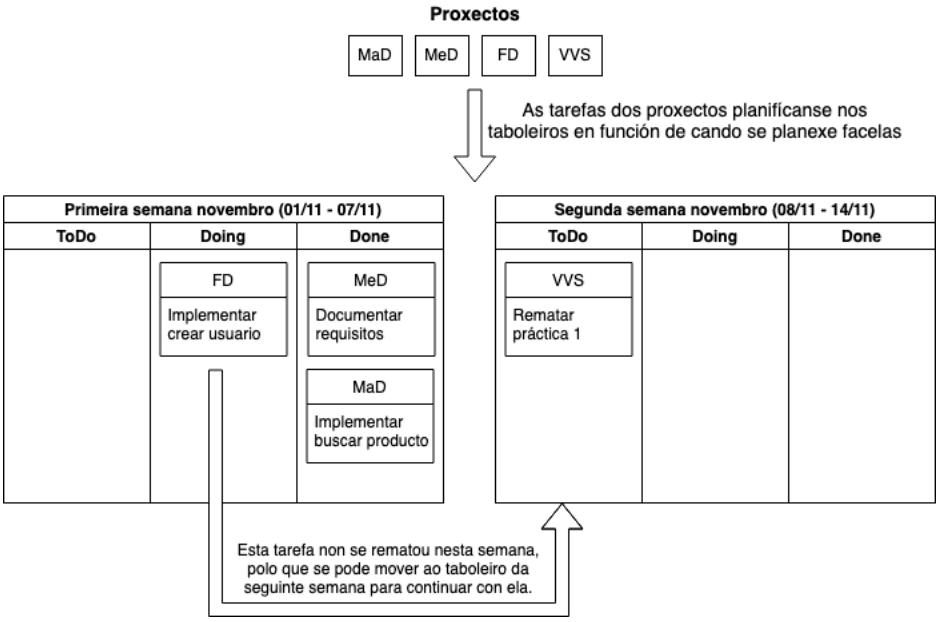

Figura 1.1: Modelado da situación dun estudante do grao

Este mesmo modelo é xeral e tamén aplicaría a outras situacións. Véxase por exemplo o caso dunha persoa que traballe e que no seu tempo libre queira obter o certificado B2 de inglés, reformar a súa casa e aprender a programar. Na figura 1.2 amósase cómo esta aplicación se adaptaría á situación desa persoa.

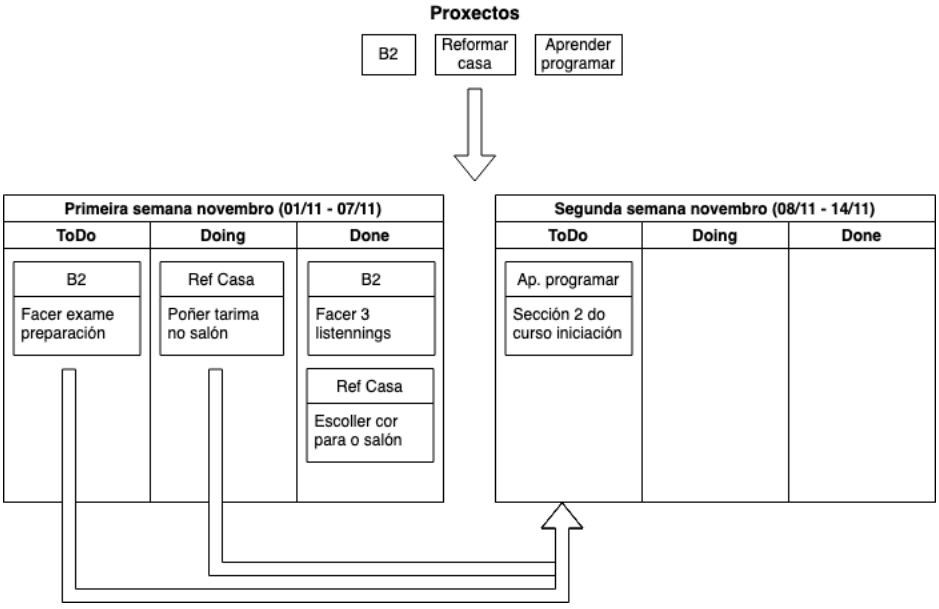

Figura 1.2: Modelado da situación con proxectos persoais

Por último, como esta aplicación pretende servir de apoio no día a día dos usuarios, buscarase crear unha interface de usuario sinxela, áxil e intuitiva para proporcionar unha experiencia de uso agradable.

1.3 Visión global do sistema

O sistema a desenrolar trátase dunha aplicación web moderna formada por un backend e un frontend. Na figura 1.3 pódese ver un diagrama da arquitectura do sistema.

O backend da aplicación, o cal contén a lóxica de negocio desta, consiste nunha API REST implementada usando tecnoloxías Java, xunto co framework Spring Boot, persistindo os datos necesarios en MySQL.

O frontend da aplicación consiste nunha aplicación web SPA implementada empregando JavaScript, apoiándose en librerías como React, Redux ou Material-UI.

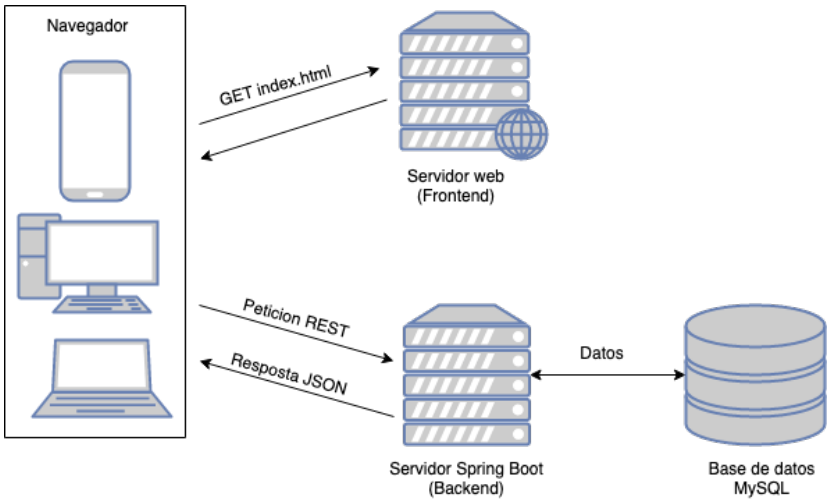

Figura 1.3: Arquitectura do sistema

Capítulo 2

Estado da arte

A xestión do tempo e das tarefas persoais sempre foi algo importante. Como consecuencia, existe un gran número de aplicacións no mercado que teñen como fin mellorar e facilitar a súa xestión.

Aquí móstranse algunhas das aplicacións do mercado que teñen funcionalidades semellantes á que se vai desenrolar neste TFG, aínda que ningunha delas as agrupa todas.

2.1 Trello

Trello [1] é unha das aplicacións máis coñecidas das baseadas en taboleiros Kanban. Trátase dunha aplicación multiplataforma que permite crear taboleiros independentes entre sí, nos que se dispoñen un número ilimitado de tarxetas nas diferentes columnas que estes conteñen.

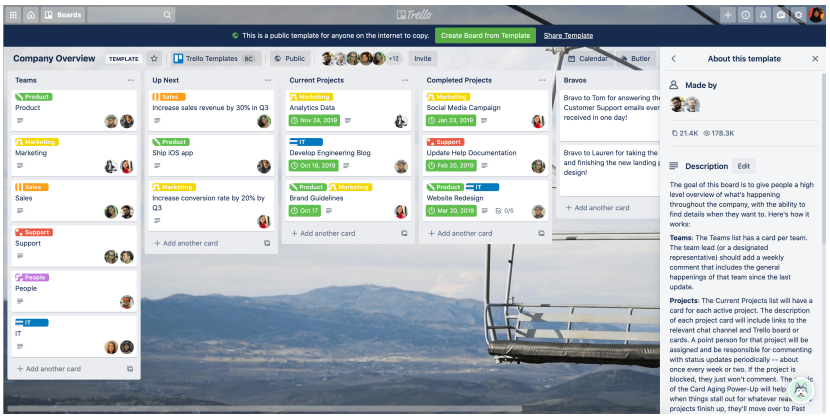

Figura 2.1: Taboleiro en Trello

Esta aplicación está máis enfocada nos fluxos de traballo colaborativos, permitindo compartir taboleiros enteiros ou tarxetas destes. Conta tamén cun sistema de comentarios nas tarxetas para que as persoas interesadas se poidan comunicar de forma sinxela, podendo manter

o foco no que trata a tarxeta.

Trello tamén conta con numerosos fluxos de automatización para os taboleiros ou as tarxetas, permitindo por exemplo crear tarxetas ou movelas de columna cando se cumple certa condición.

Esta aplicación non permite levar rexistros de tempo adicado nas diferentes tarxetas ou taboleiros. Outra característica desta é que nela tampouco existe o concepto de proxecto, que sexa independente dun taboleiro, polo tanto as tarxetas sempre están vinculadas a un único taboleiro.

2.2 Todoist

Todoist [2] é unha aplicación multiplataforma que naceu co obxetivo de permitirlle a unha persoa organizar nun único lugar todas as súas tarefas de diversos ámbitos.

Para iso, permite a creación de proxectos, nos cales se poden dispoñer as tarefas que se desexen. Para cada proxecto pódese ver as tarefas en forma de lista ou de taboleiro Kanban. Tamén conta con unha lista de tarefas diaria, na que se poden dispoñer as tarefas dos diferentes proxectos.

Todoist non permite crear taboleiros Kanban independentes dos proxectos, así como tampouco é posible levar un rexistro do tempo adicado a cada tarefa ou proxecto.

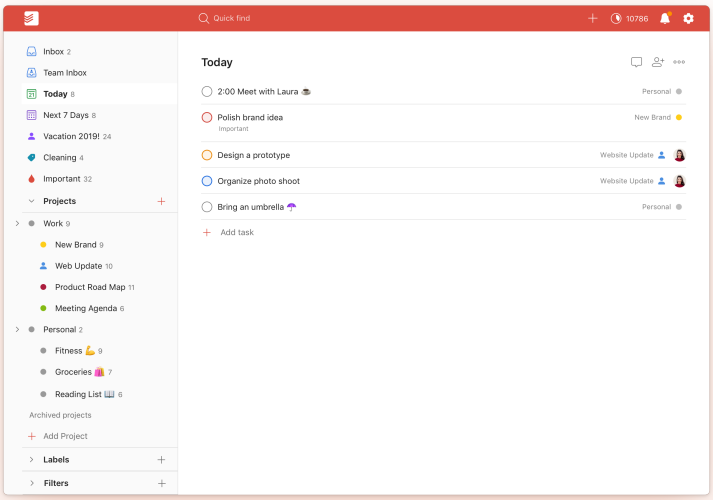

Figura 2.2: Tarefas do día en Todoist

2.3 KanbanFlow

KanbanFlow [3] é unha aplicación web que, como Trello (2.1), permite a creación de taboleiros Kanban, con cadansúas tarefas.

Esta aplicación permite levar rexistros do tempo adicado a cada tarefa, tanto usando un cronómetro, como usando a técnica Pomodoro [4]. Así mesmo, con estes datos permite ver uns informes básicos dos pomodoros realizados cada día.

Ten unha versión de pago na que ofrece informes máis interesantes do emprego de tempo, así como outros informes acerca do fluxo das tarefas no taboleiro: número de tarefas que cambian de columna por día, cantas tarefas están sin completar, etc.

Nesta aplicación tampouco existe o concepto de proxecto polo que, como en Trello, as tarefas están vinculadas a un taboleiro.

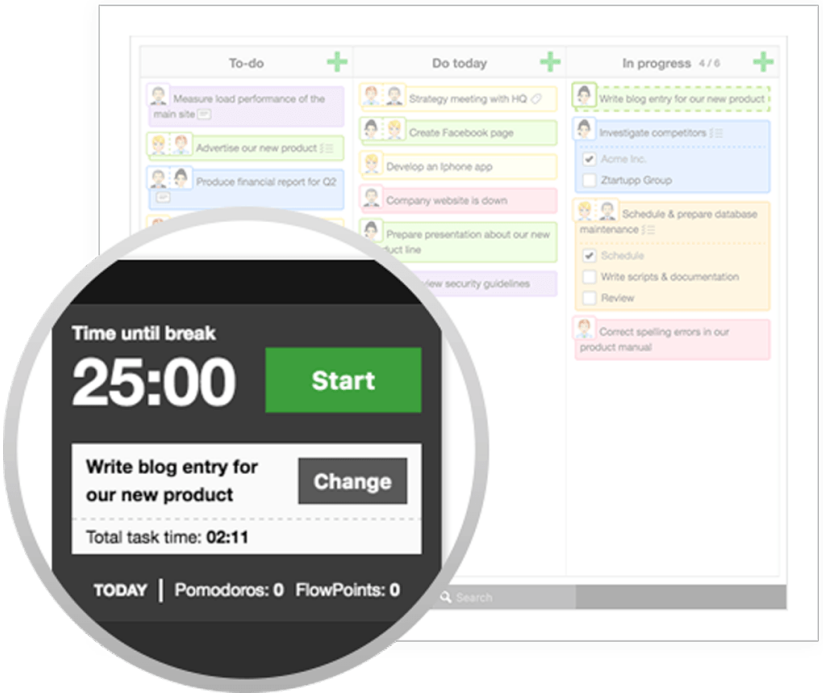

Figura 2.3: Pomodoro en kanbanflow

2.4 Outras aplicacións

Como se mencionou, existen numerosas aplicacións no ámbitos da organización de tarefas ou proxectos. Algúns outros exemplos, similares en características ás comentadas anteriormente, son:

- Microsoft To Do [5]
- 135list [6]
- Things [7]

Outra aplicación que se podería usar é Notion [8], pero esta, a diferencia das anteriores, é unha aplicación moi xenérica que permite crear documentos con táboas, listas, calendarios, taboleiros Kanban, etc. Polo tanto, permite crear espacios de traballo moi adaptables ao que necesita cada usuario. En consecuencia, un dos usos que se lle podería dar a esta aplicación é o da organización das tarefas.

Capítulo 3

Metodoloxía

Neste capítulo explicarase a metodoloxia escollida para o desenrolo do proxecto, así como a razón da súa elección e as outras alternativas contempladas.

3.1 Metodoloxía escollida

Para este proxecto escolleuse unha metodoloxia iterativa, que combina elementos das metodoloxías lineais con outros das metodoloxias áxiles.

Esta metodoloxia consiste nunha análise inicial de tódolos requisitos do sistema, para a partir deles realizar un desenrolo iterativo da aplicación, no que en cada iteración se implementará un subconxunto definido dos requisitos iniciais.

Cada iteración parte como base do produto da anterior, e para o desenrolo dos novos requisitos componse das fases de análise, deseño, implementación e probas. Unha vez rematadas estas fases, o resultado da iteración debe ser un produto que incremente a funcionalidade do da iteración anterior e que poida ser usado polos usuarios.

As principais vantaxas desta metodoloxía son:

- Os usuarios non esperan ata o final do desenrolo para usar a aplicación. Poden comezar a usala dende a primeira iteración, podendo ademais ir proporcionando retroalimentación que mellore o produto.
- Os clientes poden usar os primeiros incrementos do sistema para precisar mellor os futuros requisitos a implementar.
- Redúcese o risco de fracaso do proxecto, xa que este se divide nas diferentes iteracións deste.
- Resulta máis sinxelo adaptarse a cambios ao ter incrementos acotados.

Por outra parte, as súas principais desvantaxas son:

- En sistemas complexos, é posible que os erros nos requisitos se detecten tarde.
- Pode resultar complicado distri buir os requerimentos do sistema en iteracións, así como priorizar os que máis valor aporten para implementalos nas primeiras iteracións.

Esta metodoloxia é unha adaptación da metodoloxía Scrum [9] a este proxecto. Na seguinte sección razoarase o descarte do uso de Scrum, así como se expoñerá por que a metodoloxia aquí descrita se adapta a este proxecto.

3.2 Razoamento da elección

Inicialmente considerouse o uso de Scrum [9], xa que é unha das metodoloxías áxiles máis usadas na actualidade, que permite a entrega continuada de funcionalidades do produto, así como a posibilidade de adaptarse axilmente aos cambios. Pero descartouse o seu uso polas razóns que se expoñen a continuación.

En primeiro lugar, Scrum está claramente orientado a proxectos nos que o equipo está composto por varias persoas. Por unha parte, ten roles diferenciados que deben corresponder a diferentes persoas (Product Owner, Scrum Master). Por outra parte, establécense unha serie de reunións para mellorar a comunicación e a productividade no equipo (Daily Scrum, Sprint Retrospective, etc). No caso deste proxecto, como o equipo de desenrolo está composto por unha persoa, estes roles e reunións non serían aplicables.

A segunda razón é que en Scrum os bloques de traballo, denominados *Sprints*, teñen unha duración curta e que sempre debe ser a mesma. Isto non se adapta a este proxecto, xa que ao non poder ter unha dedicación completa a el por motivos laborais, non sería posible ter a misma dedicación de tempo en tódolos sprints.

Tras descartar a aplicación de Scrum, escolleuse unha metodoloxia iterativa que consiste no exposto na sección 3.1. A metodoloxia adáptase a este proxecto, xa que ao descartar tanto os roles como as restriccións temporais dos sprints de Scrum, xa encaixa nun equipo de unha persoa e sigue contando coa vantaxa de poder realizar entregas parciais do produto. Con respecto ás desvantaxas mencionadas, neste proxecto non supoñen un problema xa que os requisitos quedaron claramete definidos ao principio, e a súa división en iteracións tamén resultou sinxela, ao ter unha orde evidente e non ter dependencias entre eles.

Capítulo 4

Análise de requisitos global

Neste capítulo descríbense os actores que terán acceso ao sistema, así como os requisitos globais do mesmo capturados en casos de uso.

4.1 Actores

Nesta aplicación non se diferencian roles de usuario, xa que ao estar orientada ao uso persoal, un usuario autenticado xa terá acceso a tódalas funcionalidades. Polo tanto, os actores son:

- **Usuario sen autenticar:** Este actor só poderá autenticarse no sistema ou rexistrarse se non ten unha conta creada.
- **Usuario autenticado:** Este actor terá acceso a tódalas funcionalidades que ofrece a aplicación.

4.2 Casos de uso

A continuación expóñense os casos de uso da aplicación, incluindo unha breve descrición para cada un, e mockups en algúns nos que se considera relevante.

- **CU-01 Rexistro de usuario:** Un novo usuario poderá rexistrarse no sistema introducindo o seu email, nome, apelidos e un contrasinal. Unha vez rexistrado con éxito, o usuario terá acceso ás funcionalidades do sistema.
- **CU-02 Autenticación de usuario:** O usuario deberá introducir o seu email e o seu contrasinal para autenticarse no sistema. Unha vez autenticado, o usuario terá acceso ás funcionalidades da aplicación.

- **CU-03 Peche de sesión:** Un usuario autenticado poderá pechar a súa sesión. A partir dese momento só terá acceso ás páxinas de rexistro e autenticación.

- **CU-04 Ver lista proxectos:** Un usuario poderá ver os proxectos que ten actualmente creados. Para cada un mostrarase o seu nome, as datas de inicio e fin, o seu porcentaxe de completitude, a cor asociada ao proxecto e as horas que se lle leva dedicado.

  Na figura 4.1 móstrase o mockup da vista dos proxectos, coas diferentes posibilidades baralladas para mostrar os datos destes.

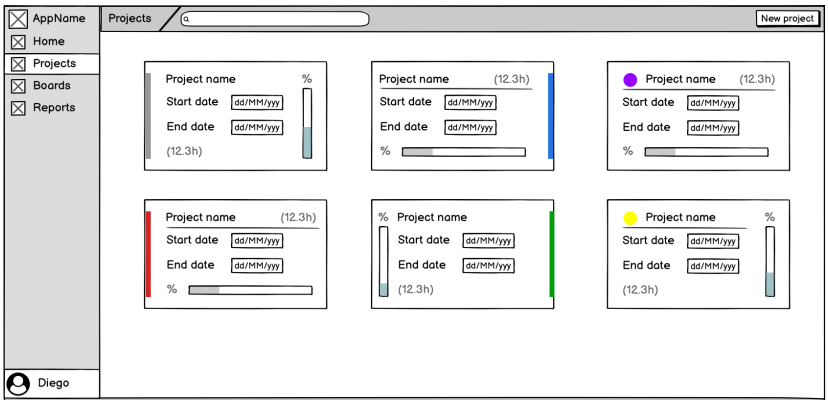

Figura 4.1: Mockup da lista de proxectos

- **CU-05 Crear proxecto:** Un usuario poderá crear un novo proxecto mediante unha ventá modal cun formulario que requerirá obligatoriamente un nome e unha cor, e opcionalmente unha descrición, unha data de inicio, unha data de fin e unha porcentaxe de completitude, que por defecto será do cero por cento.

- **CU-06 Ver detalle proxecto:** Un usuario poderá facer click nun dos proxectos da lista para ir á paxina de detalle do proxecto. Nesta páxina, na cabeceira mostraráselle o nome e a porcentaxe de completitude do proxecto. A continuación mostraránselle dúas pestanas, unha na que se ve o listado das tarefas do proxecto e outra na que se ven a descrición e as datas de inicio e fin, así como os informes relacionados co proxecto (que se explicarán no caso de uso CU-31).

  Nas figuras 4.2 e 4.3 móstranse os mockups das dúas pestanas mencionadas

- **CU-07 Actualizar proxecto:** Un usuario poderá actualizar calquera dos datos que introduciu ao crear un proxecto. Abrirase unha ventá modal, na que se cargarán os datos actuais para que o usuario os poida actualizar.

- **CU-08 Eliminar proxecto:** Un usuario poderá eliminar calquera dos seus proxectos.

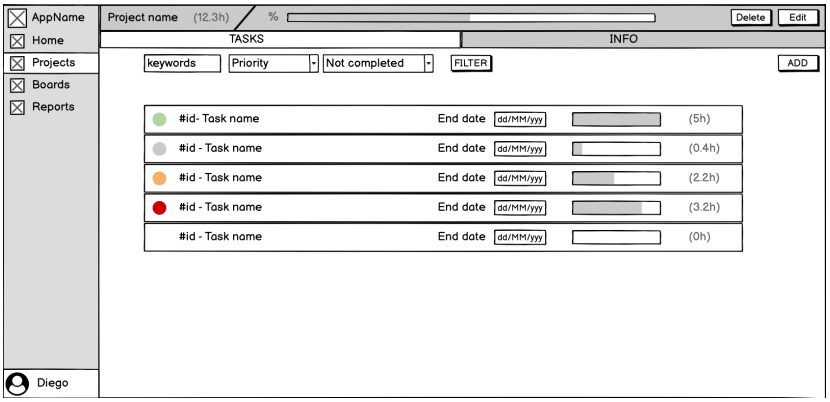

Figura 4.2: Mockup da pestana de lista de tareas dun proxecto

Figura 4.3: Mockup da pestana de información dun proxecto

- **CU-09 Filtrar tarefas proxecto:** Un usuario que está vendo o detalle dun proxecto e se atopa na pestana da lista de tarefas poderá aplicar filtros sobre esa lista. Os filtros que poderá aplicar son por palabras chave, por prioridade e por estado da tarefa (completada ou non completada).

- **CU-10 Crear tarefa proxecto:** Un usuario poderá crear unha nova tarefa para un proxecto. Para isto só será necesario que proporcione o nome da tarefa.

- **CU-11 Ver detalle tarefa:** Un usuario poderá acceder aos detalles dunha tarefa, xa sexa premendo nunha tarefa na lista nun proxecto ou nunha que se atope nun taboleiro. Despois de premer abriráselle unha ventá modal que amosa os detalles da tarefa.

  Na figura 4.4 móstrase o mockup da ventá modal de tarefa.

- **CU-12 Actualizar detalles tarefa:** Un usuario poderá actualizar a información que se lle mostra da tarefa facendo click sobre ela e actualizandoa. Nesta vista tamén poderá manexar a checklist da tarefa ou engadir subtarefas.

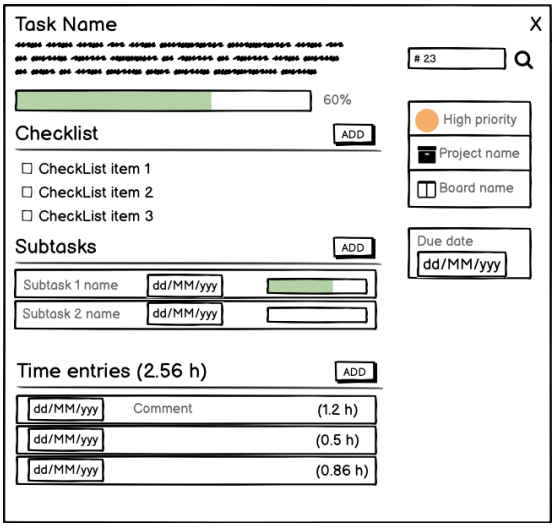

Figura 4.4: Mockup ventá modal detalle tarefa

- **CU-13 Eliminar tarefa:** Un usuario poderá eliminar calquera das tarefas dos seus proxectos.
- **CU-14 Ver lista taboleiros:** Un usuario poderá ver os taboleiros que ten actualmente creados, mostrando para cada un o seu nome e a súa data de inicio e de fin. Adicionalmente mostrarase unha etiqueta se a data de fin xa pasou.
- **CU-15 Crear taboleiro:** Un usuario poderá crear un taboleiro mediante unha ventá modal na que se mostrará un formulario que require obligatoriamente o nome e as datas de inicio e de fin do taboleiro.

  Non poderá haber intersecciónss entre os rangos temporais de dous taboleiros.
- **CU-16 Ver detalle taboleiro:** Un usuario poderá acceder á páxina de detalle dun taboleiro pinchando sobre un dos da lista. Nesta páxina mostraranse os datos do taboleiro na cabeceira. O contido da páxina formarano as columnas do taboleiro e as tarefas que conteña cada columna.

  Na figura 4.5 móstrase o mockup da vista de detalle dun taboleiro.
- **CU-17 Actualizar taboleiro:** Un usuario poderá actualizar calquera dos datos que introduciu ao crear o taboleiro.
- **CU-18 Eliminar taboleiro:** Un usuario poderá eliminar calquera dos seus taboleiros. Isto implicará que as tarefas que contiñan quedarán sen taboleiro asignado.
- **CU-19 Engadir columna a taboleiro:** Dende a vista de detalle de un taboleiro, un usuario poderá engadir unha columna a este proporcionando o nome desta.

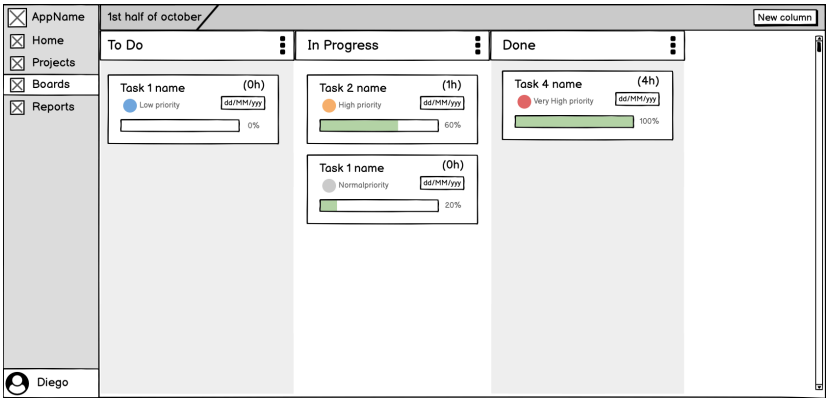

Figura 4.5: Mockup detalle taboleiro

- **CU-20 Actualizar nome columna taboleiro:** Dende a vista de detalle de un taboleiro, un usuario poderá actualizar o nome das columnas dun taboleiro.
- **CU-21 Eliminar columna taboleiro:** Dende a vista de detalle de un taboleiro, un usuario poderá eliminar calquera das columnas do taboleiro. Unha vez confirmado eliminarase esa columna e as tarefas que contiña quedarán sen taboleiro asignado.
- **CU-22 Engadir tarefa a taboleiro:** Un usuario poderá engadir unha tarefa a un taboleiro, quedando esta asignada a unha das columnas do taboleiro (por defecto engadirase na primeira columna).
- **CU-23 Eliminar tarefa de taboleiro:** Un usuario poderá eliminar unha tarefa dun taboleiro, facendo que esta quede sen taboleiro asignado.
- **CU-24 Mover tarefa a outra columna dun taboleiro:** Un usuario poderá manter pulsado sobre unha tarefa e arrastrala a outra columna do taboleiro. Unha vez deixe de pulsar, a tarefa quedará na columna sobre a que estaba arrastrando, que se mostrará resaltada cunha cor de fondo.
- **CU-25 Reordear tarefa en columna taboleiro:** Un usuario poderá manter pulsado sobre unha tarefa e arrastrala cara arriba ou cara abaixo na mesma columna. Desta forma poderá modificar a posición da tarefa nesa columna con respecto ás demais tarefas da mesma.
- **CU-26 Iniciar cronómetro nunha tarefa:** Dende a vista de detalle dunha tarefa, un usuario poderá premer no botón de *Play*, iniciando o cronómetro nesa tarefa. Unha vez iniciado, aparecerá a información do cronómetro na barra lateral esquerda da aplicación, mostrando o tempo que leva e o nome da tarefa.

Figura 4.6: Mockup da información do cronómetro

- **CU-27 Parar cronómetro:** Un usuario que ten un cronómetro activo nunha tarefa poderá premer no botón de deter. Desta forma deterase o cronómetro e abriráselle a ventá modal de engadir rexistro de tempo á tarefa, na que se cargará o tempo dese cronómetro.
- **CU-28 Crear rexistro de tempo:** Un usuario poderá engadir manualmente un rexistro de tempo a unha tarefa mediante un formulario que requerirá obligatoriamente a data (que por defecto será o día actual) e o tempo traballado, e opcionalmente un comentario.

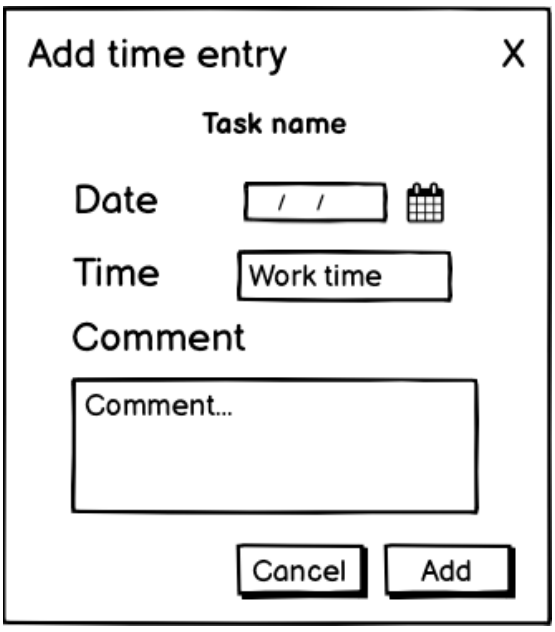

Figura 4.7: Mockup modal novo rexistro de tempo

- **CU-29 Eliminar rexistro de tempo:** Un usuario poderá eliminar calquera dos rexistros de tempo que contén unha tarefa.

- **CU-30 Ver informes globais:** Un usuario poderá acceder á sección de informes e ver os seus informes globais na pestana de informes globais. Mostraránselle tres informes:

  - IG-01 Traballo total diario: Mostrará unha gráfica circular na que se detallarán as tarefas ás que se lle dedicou tempo no día actual e o tempo dedicado a cada unha. Este informe correponde á gráfica de arriba á esquerda na figura 4.8.
  - IG-02 Traballo total por proxecto: Mostrará unha gráfica circular na que se detallará o tempo total dedicado a cada un dos proxectos. Este informe correponde á gráfica de arriba á dereita na figura 4.8.
  - IG-03 Detalle tempo de traballo por proxecto: Mostrará unha gráfica de barras na que cada barra corresponde a un día, e estará dividida por seccións correspondendo cada unha a un proxecto, e o seu tamaño ao tempo dedidado ao proxecto ese día. Este informe corresponde á gráfica de abaixo na figura 4.8.

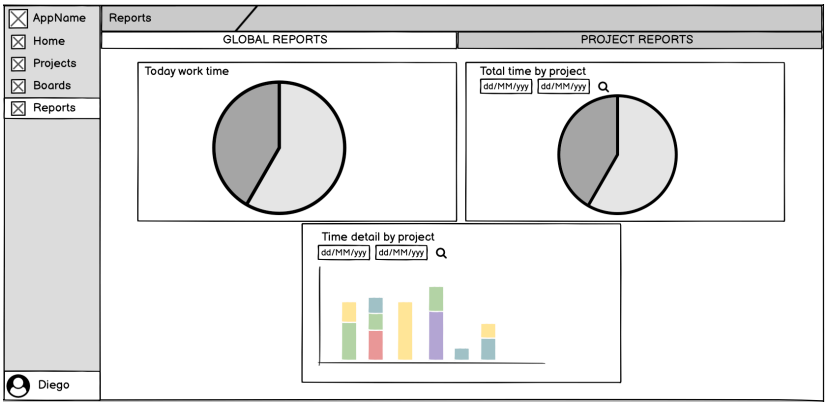

Figura 4.8: Mockup informes globais

- **CU-31 Ver informes dun proxecto:** Un usuario poderá acceder aos informes por proxecto dende dous sitios. Poderá acceder dende a pestana Info, na vista de detalle do proxecto. E tamén poderá acceder na sección de informes, dende a pestana de informes de proxecto, onde terá un desplegable para seleccionar o proxecto do que ver os informes. Mostraránselle dous informes:

  - IP-01 Estado das tarefas do proxecto: Mostrará unha gráfica circular na que se detallará o número de tarefas que se encontran sen empezar, en proceso e rematadas. Este informe correponde á gráfica de arriba á dereita na figura 4.3.
  - IP-02 Detalle tempo de traballo por tarefa: Mostrará unha gráfica de barras na que cada barra corresponde a un día, e estará dividida por seccións correspondendo

cada unha a unha tarefa, e o seu tamaño ao tempo dedidado á tarefa ese día. Este informe corresponde á gráfica de abaixo na figura 4.3.

Na figura 4.3 mostrábase o mockup desta vista de informes de proxecto.

- **CU-32 Filtrar informe por data:** Un usuario poderá axustar o intervalo de mostra de certos informes para consultar o período desexado. Os informes que se poderán filtrar son:
  - IG-02 Traballo total por proxecto
  - IG-03 Detalle tempo de traballo por proxecto
  - IP-02 Detalle tempo de traballo por tarefa

Capítulo 5

Planificación

Neste capítulo describirase cómo se levou a cabo a planificación do proxecto seguindo unha metodoloxia iterativa, mostrando o traballo realizado en cada unha das iteracións. Tamén se proporcionará un cálculo dos custos do proxecto.

5.1 Iteracións

O proxecto dividiuse en sete iteracións, que se describirán a continuación.

5.1.1 Iteración 0

Esta iteración ten como obxetivo a concepción do produto. Nela lévase a cabo a análise global de requisitos, para determinar os casos de uso que se implementarán e a súa división en iteracións. Como parte deste proceso tamén se fan os mockups das pantallas da aplicación para axudar na definición dos requisitos.

5.1.2 Iteración 1

Nesta iteración abórdase a formación nas tecnoloxías novas a usar, que neste caso só é a librería Material UI.

5.1.3 Iteración 2

A primeira iteración de desenrolo do proxecto ten como obxetivos a creación dos esqueletos do backend e do frontend, así como a realización dos casos de uso correspondentes á xestión dos usuarios:

- CU-01 Rexistro de usuarios
- CU-02 Autenticación

- CU-03 Peche de sesión

5.1.4 Iteración 3

Nesta iteración preténdese implementar todo o relacionado coa sección de proxectos. Isto inclue os casos de uso relacionados coa xestión dos proxectos, así como coa xestión das súas tarefas:

- CU-04 Ver lista proxectos
- CU-05 Crear proxecto
- CU-06 Ver detalle proxecto
- CU-07 Actualizar proxecto
- CU-08 Eliminar proxecto
- CU-09 Filtrar tarefas proxecto
- CU-10 Crear tarefa proxecto
- CU-11 Ver detalle tarefa
- CU-12 Actualizar detalles tarefa
- CU-13 Eliminar tarefa

5.1.5 Iteración 4

Unha vez completada a sección de proxectos, o seguinte é permitir planificar as súas tarefas. Polo tanto, nesta terceira iteración abórdanse os casos de uso referentes á sección de taboleiros:

- CU-14 Ver lista taboleiros
- CU-15 Crear taboleiro
- CU-16 Ver detalle taboleiro
- CU-17 Actualizar taboleiro
- CU-18 Eliminar taboleiro
- CU-19 Engadir columna a taboleiro
- CU-20 Actualizar nome columna taboleiro

- CU-21 Eliminar columna taboleiro
- CU-22 Engadir tarefa a taboleiro
- CU-23 Eliminar tarefa de taboleiro
- CU-24 Mover tarefa a outra columna dun taboleiro
- CU-25 Reordear tarefa en columna taboleiro

5.1.6 Iteración 5

Para esta iteración queda o referente ao rexistro de tempo nas tarefas e á xeración de informes útiles para o usuario en función destes datos. Os casos de uso a abordar son os seguintes:

- CU-26 Iniciar cronómetro nunha tarefa
- CU-27 Parar cronómetro
- CU-28 Crear rexistro de tempo
- CU-29 Eliminar rexistro de tempo
- CU-30 Ver informes globais
- CU-31 Ver informes dun proxecto
- CU-32 Filtrar informe por data

5.1.7 Iteración 6

A última iteración do proxecto dedícase á realización desta memoria. Como para a súa planificación xa se emprega a propia aplicación desenrolada, tamén se van detectando e implementando pequenas melloras na aplicación, sobre todo na interface de usuario.

5.2 Planificación temporal

A planificación temporal do proxecto foise realizando de forma independente para cada iteración, ao comezo desta.

A dedicación a este proxecto non foi total debido a motivos laborais, polo tanto, algunha iteración alongouse no tempo sin estar relacionado coas horas de traballo adicadas.

Iteración	Data inicio	Data fin	Estimación (h)	Tempo real (h)
Iteración 0	14/06/2021	20/06/2021	8	10
Iteración 1	21/06/2021	23/06/2021	5	5
Iteración 2	24/06/2021	12/07/2021	35	42.5
Iteración 3	13/07/2021	13/10/2021	80	128
Iteración 4	14/10/2021	01/12/2021	55	64
Iteración 5	02/12/2021	07/01/2022	50	58.5
Iteración 6	08/01/2022	14/02/2022	90	93
TOTAL	**14/06/2021**	**14/02/2022**	**310**	**401**

Táboa 5.1: Planificación temporal do proxecto e tempo real empregado

Pódese observar que a iteración 3 requeriu unha cantidade de tempo considerablemente maior que as demáis iteracións de desenrolo. Así mesmo, esta iteración foi a que máis se desviou con respecto á estimación realizada. Isto debeuse a que se tivo que dedicar máis tempo do previsto aos casos de uso correspondentes a ver e actualizar os detalles dunha tarefa, sobre todo na parte da interface web, para conseguir a experiencia de usuario desexada.

5.3 Cálculo de custos

A continuación preséntanse os custos asociados a este proxecto, dende as horas empregadas ata o custo de certos recursos necesarios. Posteriormente, na táboa 5.2 calcularase o custo total do proxecto.

Para o cálculo do custo das horas de traballo, tómase como referencia un importe de 30 euros/hora.

Custo das horas traballadas = 401 horas * 30 euros/hora = **12.030 euros**

A maiores, hai que incluír algunhos custos de recursos necesarios para poder levar a cabo este proxecto. Ao non estar a tempo completo dedicado a este, para calcular a súa duración nun caso real divídense as horas totais entre as horas dun mes a tempo completo. Polo tanto:

Duración real = 401 horas / 160 horas/mes = **2.51 meses**

A duración do proxecto nun caso real sería de aproximadamente dous meses e medio. Os custos dos recursos necesarios para levar a cabo o proxecto foron:

- **Equipos informáticos:** Adquiriuse un ordenador portátil, un monitor externo, un teclado e un rato por un custo total de **1.150 euros**. Como estes equipos se poderán utilizar en proxectos futuros, a este proxecto impútaselle o 30% do seu custo total. Polo tanto, son **345 euros**
- **Oficina:** Alquílase unha pequena oficina por un custo mensual de **180 euros**, resultando un custo total de **540 euros** (3 meses).
- **Electricidade:** Tivo un custo total de **90 euros** (3 meses).
- **Internet:** Contratouse unha tarifa básica de datos, por **19 euros ao mes**, polo tanto en total foron **57 euros** (3 meses).

Na seguinte táboa amósase o cálculo do custo total do proxecto.

Concepto	Custo (euros)
Horas traballadas	12.030
Equipos informáticos	345
Alquiler oficina	540
Electricidade	90
Internet	57
Custo total do proxecto	**13.062**

Táboa 5.2: Cálculo do custo total do proxecto

O custo total do proxecto foi de **13.062 euros**.

Capítulo 6

Fundamentos tecnolóxicos

6.1 Tecnoloxías empregadas no backend

6.1.1 Java

Java é unha linguaxe de programación de propósito xeral, con tipado forte, concurrente e orientada a obxetos. Permite o desenrolo de aplicacións independentes da plataforma na que se pretende executar, grazas ao Java Runtime Environment (JRE), que permite executar o mesmo código fonte compilado nos diferentes entornos, sempre que estes conten co JRE.

6.1.2 Spring Boot

Spring Boot [10] é unha das principais ferramentas no ecosistema de desenrolo web con Java. Trátase dun framework de código aberto que naceu co obxetivo de simplificar ao máximo a configuración necesaria para executar un proxecto. Algunhas vantaxas que aporta Spring Boot son:

- Inversión de control e inxección de dependencias
- Desplegue áxil da aplicación nun servidor web embebido (Tomcat, Jetty, etc.)
- Proporciona diferentes grupos de dependencias relacionadas entre si, co obxetivo de simplificar o manexo das dependencias do proxecto, aos que chaman starters
- Non se necesitan os arquivos XML de configuración que utiliza Spring framework

6.1.3 Maven

Maven [11] é unha ferramenta empregada para a construcción e xestión de proxectos Java. Proporciona un modelo estándar de definir proxectos, o Project Object Model (POM),

que é un ficheiro XML no que se describen as características do software a costruir, as súas dependencias, plugins, etc.

Unha vez configurado o POM, Maven permite xestionar o proxecto ao completo, dende a compilación do código, pasando polos tests, ou o empaquetado e desplegue. Todo isto se consegue co ciclo de vida de Maven, que se compón dunha serie de etapas fácilmente configurables.

6.1.4 MySQL

MySQL [12] é un sistema xestor de bases de datos de código aberto basado en SQL. É o máis extendido e utilizado pola súa seguridade, velocidade e escalabilidade.

6.1.5 Eclipse

Eclipse [13] é un Integrated Development Environment (IDE) que permite o desenrolo en varias linguaxes, e que foi concebido dende o inicio para ser extendido de forma indefinida mediante plugins. Grazas a esto proporciona un gran número de funcionalidades que axudan ao programador, así como permite integrarse facilmente con outras ferramentas usadas no desenvolvemento.

6.1.6 Postman

Postman [14] permite, entre outras cousas, o envío de peticións HTTP a APIs REST, polo que é de especial utilidade para realizar probas sobre esta.

6.2 Tecnoloxías empregadas no frontend

6.2.1 JavaScript

JavaScript é unha linguaxe interpretada (non necesita ser compilada previamente), multiparadigma, dinámica e baseada en prototipos. Soporta programación funcional (declarativa), orientada a obxetos ou imperativa.

6.2.2 React

React [15] é unha librería de JavaScript que pretende facilitar o desenvolvemento de interfaces de usuario interactivas de unha sola páxina (SPA). A construcción de aplicacións con React lévase a cabo de forma declarativa e creando compoñentes reusables.

Un compoñente é unha función JavaScript que React utiliza para ”pintar” unha parte da interface. Estes compoñentes poden manexar un estado propio e variar a súa apariencia en

función de ese estado. Así mesmo, un compoñente pode estar formado por outros compoñentes, permitindo construir interfaces de usuario en base a moitos compoñentes reusables.

Estes compoñentes permiten usar JSX, que é unha extensión da sintaxe de Javascript que en React se usa para describir como se verá a interface de usuario. JSX permite escribir etiquetas semellantes a como sería en HTML, pero permitindo insertar código JavaScript poñendo este entre chaves. A continuación móstrase un exemplo de un compoñente que usa JSX.

```
const HelloWorld = ({ name }) => {
    return <div>Hello world! This is {name}!</div>;
}
```

Outra característica importante de React é que as actualizacións dos compoñentes non se realizan directamente sobre o DOM do navegador. Isto débese a que actualizar este DOM é unha tarefa custosa, polo que en aplicacións nas que houbera bastantes actualizacións dos compoñentes a experiencia do usuario veríase afectada. Para solucionar isto, React conta co Virtual DOM, que non é máis que unha réplica exacta do DOM do navegador, pero moito máis eficiente. Polo tanto, os compoñentes de React actualízanse neste DOM virtual, e posteriormente React calcula as diferencias deste co DOM do navegador e as actualiza nel, conseguindo actualizacións de estado moito máis eficientes.

6.2.3 Redux

Redux [16] é un contedor do estado de aplicacións JavaScript que se pode usar xunto con React ou con calquera outra librería. Con Redux extráese gran parte do estado dos compoñentes a un único obxeto, denominado ”store”. O estado almacenado neste obxeto é de só lectura, e para cambialo definense as accións. Estas accións son obxetos planos que se invocan cando se desexa actualizar o estado, e son procesadas por unha función especial, chamada ”reducer” na que se define cómo cada acción afecta ao estado da aplicación.

6.2.4 Material UI

MUI [17] é unha librería de código aberto que proporciona un gran número de compoñentes para usar con React. Estes compoñentes seguen as guías de Material Design.

Material Design [18] é un sistema de diseño creado por Google para axudar na construcción de interfaces de usuario atractivas e fácilmente adaptables a plataformas web e móviles.

6.2.5 Yarn

Yarn [19] é un xestor de dependencias alternativo a NPM no entorno de NodeJS. Por defecto usa o rexistro de de dependencias de NPM, polo que non hai que realizar configuración

adicional. Está enfocado na seguridade e na velocidade, e tamén ofrece un feedback bastante amigable para o usuario.

6.2.6 VS Code

Visual Studio Code [20] é un editor de código desenvolto por Microsoft. É de código aberto e está desenrolado usando o framework electron [21], polo que se executa nun navegador web embebido. De base xa inclúe soporte de resaltado de sintaxis, autocompletado ou depuración para un bo número de linguaxes e, como eclipse, permite extender a súa funcionalidade mediante plugins, facendo posible que soporte outras linguaxes, integralo con outras ferramentas usadas, etc.

6.3 Tecnoloxías complementarias

6.3.1 Git

Git [22] é un sistema de control de versións que facilita o seguemento dos cambios nos ficheiros dun proxecto. Trátase do SCM máis usado na actualidade. É distribuido, polo que cada usuario posúe unha copia local do repositorio, sobre a que fai os cambios antes de subilos ao repositorio remoto e xuntalos cos dos outros desenvolvedores.

6.3.2 Docker

Docker [23] é un proxecto de código aberto que automatiza o desplegue de aplicacións dentro de contedores de software. Isto proporciona una capa adicional de abstracción que fai ó contenedor funcionar independentemente do sistema operativo no que se execute. Estes contedores créanse a partir de imaxes, que son unha especie de plantillas que conteñen o que se executará, por exemplo un sistema operativo ou unha aplicación web.

Empregouse Docker, en primeiro lugar para desplegar áxilmente un contedor con MySQL para usar no desenrolo da aplicación. E en segundo lugar, para facilitar o desplegue desta, xa que simplemente se crea e configura un ficheiro chamado docker-compose.yaml empregando as imaxes que se xeran do frontend e do backend, xunto con unha de MySQL. Ao executar o docker-compose, os tres contedores levántanse e a aplicación xa está funcionando.

6.3.3 KanbanFlow

KanbanFlow xa foi explicado na sección 2.3. Usouse esta ferramenta para crear e rexistrar o tempo nas diferentes tarefas deste proxecto, así como para dispoñelas de forma visual nun taboleiro Kanban.

Na última iteración do proxecto (5.1.7) xa se empregou a propia aplicación desenrolada.

Capítulo 7

Desenvolvemento

Neste capítulo detallaranse os aspectos máis relevantes do proceso de desenvolvemento da aplicación, tanto técnicos como funcionais.

En primeiro lugar describirase a estrutura da aplicación e o seu modelo de datos. A continuación farase un resumo das iteracións de desenrolo levadas a cabo no proxecto.

7.1 Estrutura da aplicación

7.1.1 Estrutura do backend

Como se mencionou, o backend é un proxecto maven, e componse dos seguintes ficheiros e directorios:

- **/src/main/java:** Neste directorio está o código fonte do backend.
  - **model:** Paquete no cal se atopa a estructura de paquetes e clases que conteñen as capas de acceso a datos e de lóxica de negocio do backend. A capa de acceso a datos está formada polas entidades e os DAO. A capa de lóxica de negocio contén os servizos e as excepcións que lanzan estes.
  - **rest:** Paquete no cal se atopa a capa de servizos do backend. Contén os controladores REST, os DTO usados e tamén a configuración da seguridade en Spring.
  - **Application.java:** Clase raíz dunha aplicación que usa Spring Boot e que permite arrancar esta.
- **/src/main/resources:** Este directorio contén o ficheiro application.yml, o cal contén a configuración do backend. Neste directorio tamén se atopan os ficheiros de internacionalización para as mensaxes de erro devoltas polo backend.
- **/src/test/java:** Contén tódalas clases dos test realizados sobre o código do backend.

- **/src/test/resources:** Aquí atópase o ficheiro application.yml que contén a configuración específica do backend para os tests.
- **docker-compose.yaml:** Este ficheiro configura os contedores docker cos servizos que se utilizarán no proceso de desenvolvemento da aplicación. Neste caso levanta un contedor con MySQL empregando o comando docker-compose up.
- **pom.xml:** Como se trata dun proxecto Maven, existe este ficheiro para almacenar a configuración do proxecto.

7.1.2 Estrutura do frontend

O código fonte do frontend atópase dentro do directorio /src. A parte deste, tamén contén algún outro directorio e ficheiro necesarios. A continuación móstrase en detalle:

- **/public:** Directorio que contén o logo, icono e imaxes que se usan na app, así como o único ficheiro .html desta, o index.html.
- **/src/backend:** Contén a lóxica de realización das consultas á API REST do backend.
- **/src/components:** Almacénanase tódolos compoñentes de React usados no frontend. Están ordenados en diferentes directorios: UI, Layout, reports, projects, tasks, etc.
- **/src/helpers:** Contén ficheiros con lóxica reusable entre distintos compoñentes.
- **/src/hooks:** Contén algúns hooks propios creados para agrupar lóxica reusable entre compoñentes e que emprega hooks de React.
- **/src/i18n:** Contén os arquivos de internacionalización.
- **/src/store:** Aquí atópase toda a lóxica correspondente a Redux para manexar o estado do frontend. Conten un ficheiro co Reductor base, e a continuación atópanse diferentes directorios (boards, projects, etc.) nos que cada un contén as accións para modificar o estado, os selectores para acceder ó estado e o Reductor dese módulo.
- **/src/index.js:** Renderiza no DOM o compoñente base App.js do frontend. Así mesmo, aquí inicialízase o store de Redux, defínese o tema de MUI e configurase a internacionalización.
- **Dockerfile:** Ficheiro necesario para xerar a imaxe Docker que contén o frontend da aplicación
- **package.json:** Ficheiro que almacena a configuración do proxecto.

7.2 Modelo de datos

Na figura 7.1 móstrase o diagrama de entidades da aplicación, que representa o seu modelo de datos. A continuación realizanse unhas breves aclaracións sobre este diagrama.

A entidade máis relevante é a tarefa (Task), que debe estar nun proxecto (Project) e pode estar ou non nunha columna (BoardColumn). Así mesmo, unha tarefa pode conter subtarefas ou elementos da checkList (CheckListItem). Para levar a conta do tempo dedicado, unha tarefa pode ter rexistros de tempo (TimeEntry).

Por outra parte, un usuario pode ter proxectos e taboleiros (Board).Un usuario tamén pode ter un cronómetro (TrackerInfo), cando ten un cronómetro activo. Este cronómetro debe estar relacionado cunha tarefa.

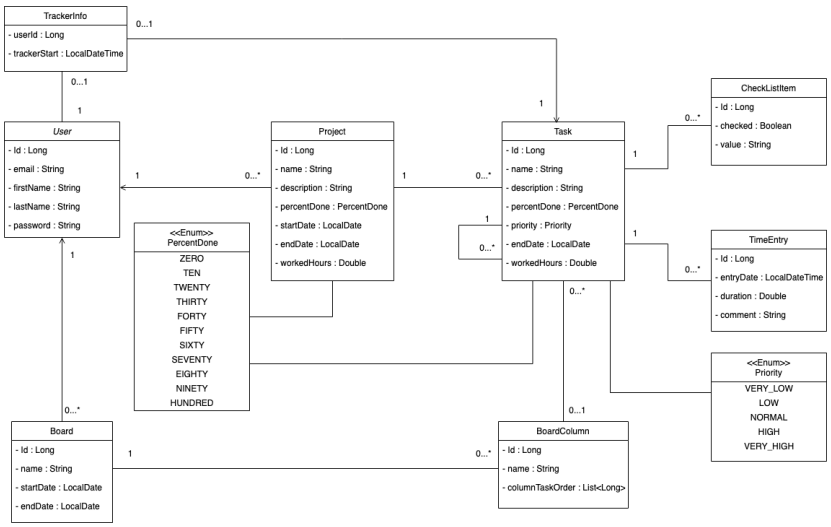

Figura 7.1: Diagrama de entidades da aplicación

7.3 Iteración 1

Nesta primeira iteración levouse a cabo a formación en MUI. Para isto seguíronse os exemplos e documentación que proporciona MUI na súa web. Así mesmo, probáronse certos exemplos nun entorno local para entendelos mellor, e entender ben tamén o funcionamento de MUI e do seu sistema para dar estilos aos compoñentes, mediante o uso dun tema.

7.4 Iteración 2

7.4.1 Análise

A primeira iteración de desenrolo do proxecto ten como obxetivos, en primeiro lugar a creación dos esqueletos do backend e do frontend. A continuación realizanse os casos de uso correspondentes á xestión dos usuarios. Finalmente constrúese a páxina base á que acceden os os usuarios autenticados.

O resultado desta iteración é unha aplicación que permite a xestión de usuarios e que mostra unha páxina base aos usuarios autenticados, preparada para engadirlle as funcionalidades das seguintes iteracións. Os casos de uso abordados foron:

- CU-01 Rexistro de usuarios
- CU-02 Autenticación
- CU-03 Peche de sesión

7.4.2 Deseño e implementación

Nesta iteración comezouse pola creación dos esqueletos do backend e o frontend para constituír a base a partir da cal se desenvolverían os casos de uso da aplicación. A estrutura dos proxectos mostrouse na sección 7.1.

CU-01 Rexistro de usuarios

Para poder rexistrar usuarios, comezouse coa parte do backend, en primeiro lugar creando a entidade de usuario (User), o seu DAO e o servizo de usuarios UserService, que contén un método coa lóxica correspondente á creación do usuario. A continuación creouse un controlador REST (UserController) habilitando nel unha operación POST para o rexistro de usuarios. Esta operación recibe tódolos datos do usuario e invoca o método do servizo que, tras validar que non exista un usuario co mesmo email, engade o novo usuario na base de datos.

No frontend comezou por crearse a petición fetch que chama ao backend. A continuación creouse o reducer de Redux para a parte de usuarios, creando tamén a acción correspondente para rexistrar un usuario. Para a parte da interfaz creouse un compoñente ***SignUp***, que contén un formulario con cinco ***TextField*** de MUI, correspondentes a: email, nome, apelidos, contrasinal e confirmación do contrasinal. Na figura 7.2 móstrase a vista deste compoñente.

Este compoñente é un controlled component, o que quere decir que mantén un estado privado e interno ao compoñente, independente do resto da aplicación. Este

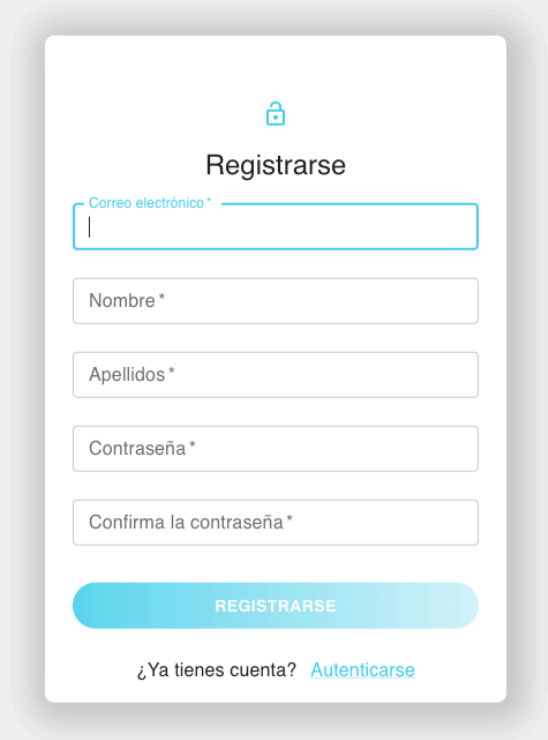

Figura 7.2: Vista do compoñente de rexitro de usuario

tipo de compoñentes acostuman a usarse para os formularios, permitindo almacenar os valores de cada campo e tamén dando a posibilidade de facer validacións dos datos introducidos para así dar feedback ao usuario se introduce datos erróneos.

Para evitar duplicidade de código e para agrupar un conxunto de lóxica reusable nun só punto creouse un hook useInput(), que agrupa a lóxica da xestión de estado dun input, así como da a posibilidade de realizar validacións sobre este valor a medida que vai cambiando, mediante o parámetro que recibe chamado *valueIsValid*, que é unha función que se lle pasa para que realice a validación. O feito de pasarlle esta función por parámetro fai posible que este mesmo hook poida manexar inputs con calquera tipo de validación requerida. Este hook reusase en casi a totalidade dos inputs do frontend, a excepcións de algúns que presentaban algunha peculiaridade. A continuación móstrase o código do hook:

```
const useInput = (valueIsValid, initialValue = "") => {
  const [inputValue, setInputValue] = useState(initialValue);
  const [isTouched, setIsTouched] = useState(false);
  const [isValid, setIsValid] = useState(true);

  useEffect(() => {
```

```
    const timeout = setTimeout(() => {
      isTouched &&
        setIsValid(valueIsValid
            ? valueIsValid(inputValue)
            : true
        );
    }, 500);
    return () => {
      clearTimeout(timeout);
    };
  }, [isTouched, inputValue, valueIsValid]);

  const inputChangeHandler = (newValue) => {
    isTouched || setIsTouched(true);
    setInputValue(newValue.trimStart());
  };

  return {
    value: inputValue,
    isValid,
    inputChangeHandler,
  };
};

export default useInput;
```

CU-02 Autenticación

Para a implementación da autenticación de usuarios optouse polo uso do estándar JSON Web Token (JWT) [24], xa que desta forma o backend non ten que almacenar ningún tipo de estado entre peticións, ao ser o cliente o que entrega o token en cada petición que realiza.

Para autenticarse habilitouse unha operación POST no backend. Esta operación recibe un email e un contrasinal, valida que as credenciais coinciden coas almacenadas na base de datos e despois xera un token que formará parte da resposta do backend, e que o cliente deberá usar nas seguintes peticións á API REST do backend.

No frontend, este token almacénase no estado de Redux, para poder envialo en cada petición. Tamén se garda no almacenamento local do navegador, xa que desta forma se o usuario recarga a páxina poderase recuperar de ahí o token, polo que non terá que volver a autenticarse.

Na figura 7.3 móstrase un diagrama de secuencia no que se detalla o proceso de autorización para unha petición usando JWT.

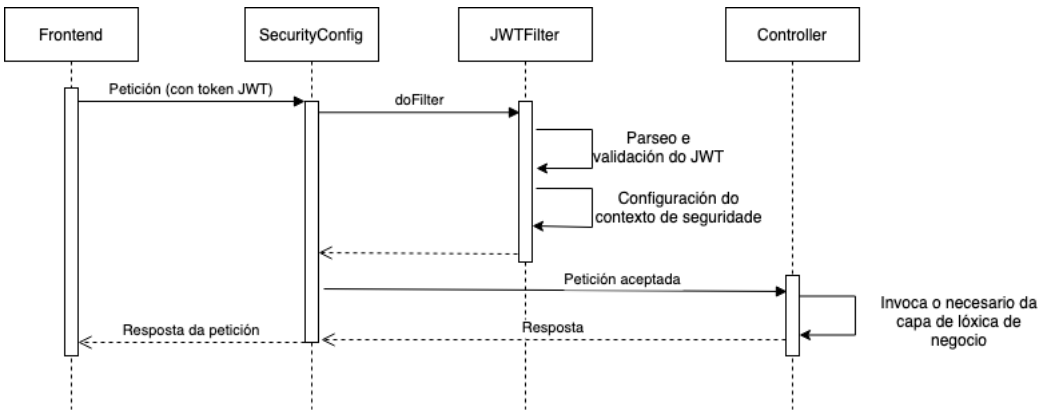

Figura 7.3: Diagrama de secuencia autorización con JWT

Na parte do frontend creouse tanto a chamada ó backend como a acción necesaria en Redux. A continuación creouse un compoñente ***Login***, que contén un formulario con dous ***TextField***, un para o email e outro para o contrasinal. Na figura 7.4 móstrase a vista deste compoñente.

Figura 7.4: Vista do compoñente de login de usuario

Despois disto xa era posible rexistrar e autenticar usuarios, polo tanto o seguinte paso foi a creación da vista base á que accederían os usuarios autenticados. Na figura 7.5 pódese ver a vista desta páxina, da que se detallan a continuación os seus compoñentes máis relevantes:

- **ApplicationLayout:** Compoñente raíz do Layout de usuarios autenticados. Contén a barra lateral esquerda (LeftSidebar) e o Router que se encarga de mostrar o compoñente adecuado en función da ruta.

- **LeftSidebar:** Representa a barra lateral esquerda da aplicación.
- **SidebarItem:** Elemento da barra lateral que enlaza a unha sección da aplicación.
- **UserProfile:** Contido na barra lateral esquerda da aplicación, mostra o nome do usuario e se pincha nel mostra as opcións deste (por exemplo, a de saír).
- **Page:** Compoñente base para usar nos compoñentes que irán nas rutas do Router (páxinas).
- **Header:** Está contido no compoñente Page, este é un compoñente que se usa para mostrar o título da páxina así como certas accións propias de cada unha.
- **PageContentContainer:** Está dentro do compoñente Page, e este compoñente define unha serie de estilos base para mostrar despois o contido das páxinas.

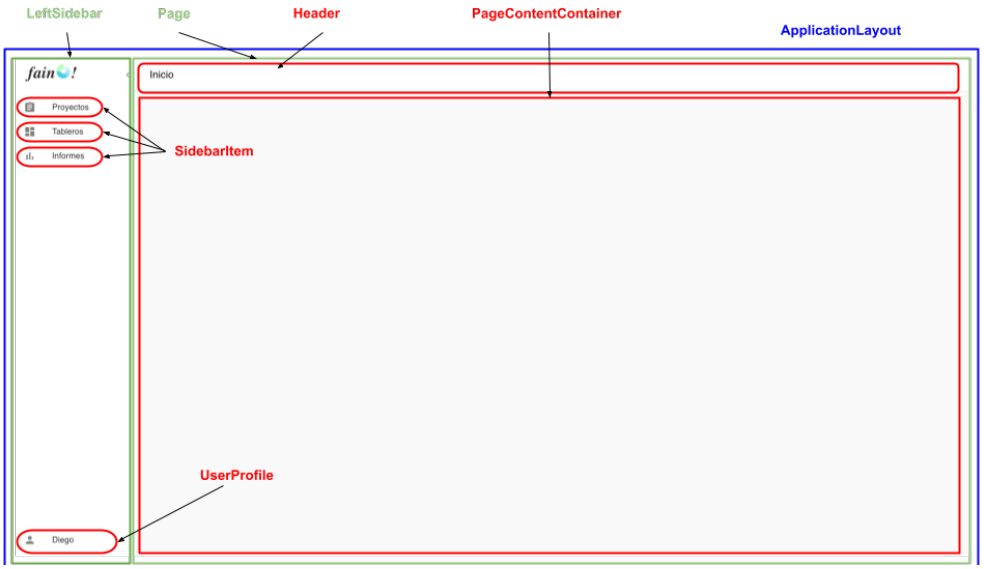

Figura 7.5: Vista base dos usuarios autenticados

CU-03 Peche de sesión

Como o backend non almacena ningún tipo de estado de autenticación, para cerrar a sesión só hai que eliminar o token JWT do estado de Redux e do Local Storage. Polo tanto, como non se realiza ningunha chamada ó backend, para implementar esta función só foi necesario definir un novo tipo de acción para levar a cabo a operación. Esta acción dispárase cando o usuario preme o botón *Sair*, situado no desplegable que aparece ao facer click sobre o seu nome ***(UserInfo)***, na barra lateral esquerda.

7.5 Iteración 3

7.5.1 Análise

O obxetivo deste iteración é o de implementar todo o relacionado coa sección de proxectos. Desta forma, o resultado será unha aplicación que, a maiores da anterior iteración, permitiralle a un usuario definir os seus proxectos. Para cada proxecto xa poderá definir tódalas tarefas necesarias cun bo nivel de detalle. O usuario tamén podería ir rexistrando avances nas tarefas ou proxectos, xa que podería ir actualizando as súas porcentaxes de completitude ou tamén marcar elementos das checklist das tarefas. Os casos de uso a abordar son:

- CU-04 Ver lista proxectos
- CU-05 Crear proxecto
- CU-06 Ver detalle proxecto
- CU-07 Actualizar proxecto
- CU-08 Eliminar proxecto
- CU-09 Filtrar tarefas proxecto
- CU-10 Crear tarefa proxecto
- CU-11 Ver detalle tarefa
- CU-12 Actualizar detalles tarefa
- CU-13 Eliminar tarefa

7.5.2 Deseño e implementación

Nesta iteración comezou por deseñarse e crear a estrutura de clases do backend sobre a que se implementarían as funcionalidades relacionadas con proxectos e tarefas. Na figura 7.6 móstrase o diagrama de clases. Nel pode observarse que existen dous controladores, encargados de expoñer o endpoint REST, que utilizan cadanseu servizo, no cal reside a lóxica de negocio. Estes servizos utilizan un servizo común *UserChecker*, que se usará para ter lóxica reusable sobre validar que un usuario exista ou que un proxecto pertenza a un usuario. Tamén mencionar a existencia dos DAO

de paxinación, os cales se impleméntan no backend para realizar unha implementación propia da paxinación. Ademais, no *TaskPaginationDao* tamén residirá a lóxica de filtrado de tarefas.

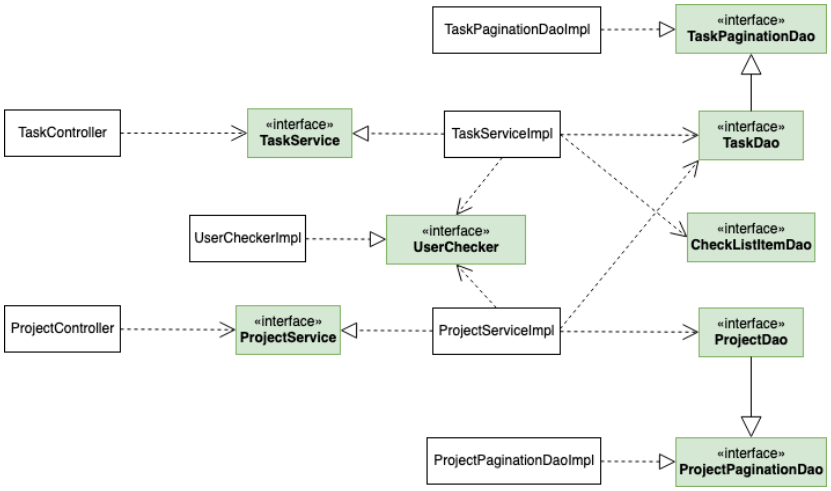

Figura 7.6: Diagrama de clases encargadas das operacións sobre proxectos e tarefas

CU-05 Crear proxecto

Para a creación dun proxecto implementouse unha operación POST no backend que recibe os datos do proxecto no corpo da petición e o crea se o usuario non ten outro co mesmo nome. No frontend, en primeiro lugar, engadiuse un compoñente ***Projects***, que será usado despois para mostrar a lista de proxectos. Pero polo momento só se creou, usando o compoñente ***Page*** e se lle engadiu na cabeceira un botón para permitir crear proxectos, o cal abre un compoñente ***CreateProjectModal***. Este utiliza un compoñente ***ProjectFormModal*** que será reutilizado posteriormente no caso de uso de editar un proxecto. Este compoñente mostra unha ventá modal cun formulario composto por tres ***TextField***, para nome, descrición e porcentaxe de completitude (neste último caso ao compoñente hai que pasarlle unha prop *select* como true para que funcione como un selector). A maiores, no formulario utilízanse dous ***DesktopDatePicker*** para as datas de inicio e fin e un compoñente propio para a cor.

Así mesmo, tamén se engadiu a correspondente acción en Redux e se creou neste paso o Reducer da sección de proxectos. A acción engadida sería disparada dende a modal, ao confirmar o formulario. Na figura 7.7 móstrase a vista da ventá modal de crear proxecto.

CU-04 Ver lista proxectos

Para este caso de uso optouse pola utilización dunha lista paxinada. Para isto, inicialmente implementouse no backend unha operación GET que recibe como pa-

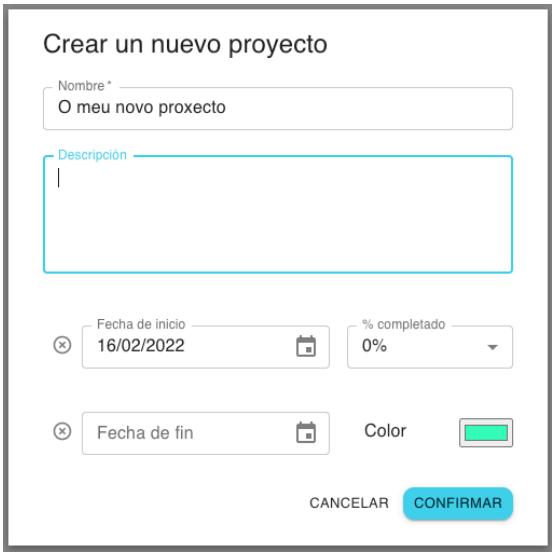

Figura 7.7: Vista da ventá modal para crear ou actualizar proxecto

rámetros a páxina a buscar e opcionalmente o número de elementos a devolver, que por defecto será de 12. A consulta paxinada á base de datos implementouse mediante o ***ProjectPaginationDaoImpl***, mostrado na figura 7.6, no cal se realiza a consulta paxinada á base de datos permitindo devolver os elementos da páxina así como o número total de elementos.

En canto ao frontend, creouse o compoñente ***ProjectsGrid***, que se engadiu como contido á páxina creada no anterior caso de uso. Este compoñente usa internamete o compoñente ***Grid*** de MUI. Úsase ese compoñente tanto para o grid contedor como para cada elemento contido neste, xa que está deseñado de forma que se lle pode pasar a prop *container* ou *item* a true en función da necesidade. Pada elemento do grid creouse o compoñente ***ProjectGridItem***. Para a información mostrada de cada proxecto foi necesario crear algún compoñente que se reusaría en bastantes máis partes da interface. Estes son ***DateDisplay***, para mostrar unha data e ***PercentDone*** para mostrar unha barra coa porcentaxe de completitude.

Para mostrar os controis da paxinación engadiuse o compoñente ***Pagination***, tamén de MUI, que permite mostrar as frechas para cambiar de páxina así como os números das páxinas, para que o usuario poida avanzar á que precise. A continuación móstrase na figura 7.8 a vista da páxina de proxectos, con algunhos proxectos de exemplo e na que tamén aparece o compoñente da paxinación.

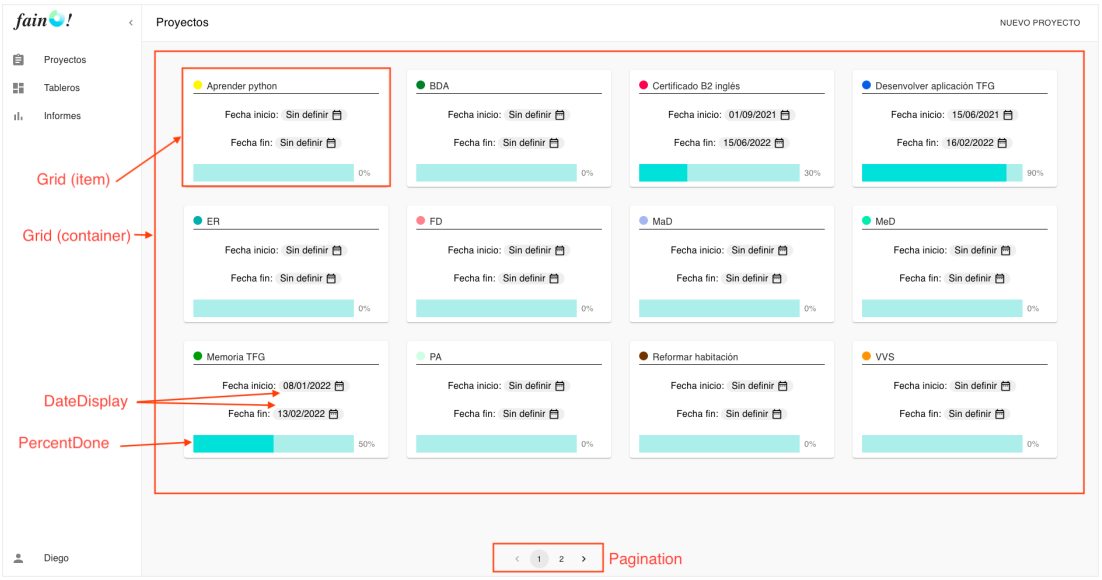

Figura 7.8: Vista da lista de proxectos

CU-06 Ver detalle proxecto

Para a implementación deste caso de uso necesitáronse dúas operacións GET no backend, unha para devolver os detalles dun proxecto buscando por ID e outra para devolver a lista de tarefas do proxecto. Para esta segunda tamén se optou por unha lista paxinada, implementada no ***TaskPaginationDaoImpl***.

No frontend creouse o compoñente ***ProjectDetails*** que tamén usa o compoñente ***Page*** e representa a páxina de detalle dun proxecto. O contido desta páxina son dúas pestanas, unha para mostrar as tarefas do proxecto e outra para mostrar información deste. Para isto úsase o compoñente ***Tabs*** propio. Este compoñente encapsula a estrutura relacionada co renderizado de pestanas que propón MUI, expoñendo simplemente unha prop na que se lle pasará unha lista das pestanas (para cada unha necesita o nome e o compoñente desa pestana). Este compoñente propio internamente utiliza os compoñentes ***Tabs***, ***Tab*** e ***TabPanel*** de MUI.

Para renderizar a primeira pestana creouse o compoñente ***ProjectTasksTab***, o cal carga as tarefas do proxecto invocando a correspondente acción de Redux. Mediante un selector accede ao estado de Redux para obter esta lista, polo que cando as tarefas se cargan no estado, estas renderízanse empregando o compoñente ***ProjectTasksTable***. Este compoñente mostra as tarefas usando compoñentes de MUI. Renderiza unha tabla, mediante o compoñente ***Table***, a cal ten unha cabeceira ***TableHead*** composta por unha fila ***TableRow*** e varias celdas nas que vai o nome de cada unha. A continuación ten un corpo ***TableBody*** no que se mostra por cada tarefa o compoñente

propio ***TaskTableItem***. Internamente este compoñente é unha ***TableRow*** que contén as celdas correspondentes. Nalgunhas destas celdas, para mostrar o seu contido, empréganse certos compoñentes reusables anteriormente creados (***DateDisplay*** ou ***PercentDone***). Tamén se creou un novo compoñente reusable para mostrar a prioridade (***Priority***). Ao final da táboa atópase o compoñente ***TablePagination*** de MUI, que permite cambiar o número de resultados obtidos e navegar entre as páxinas. Na figura 7.9 móstrase a vista desta pestana.

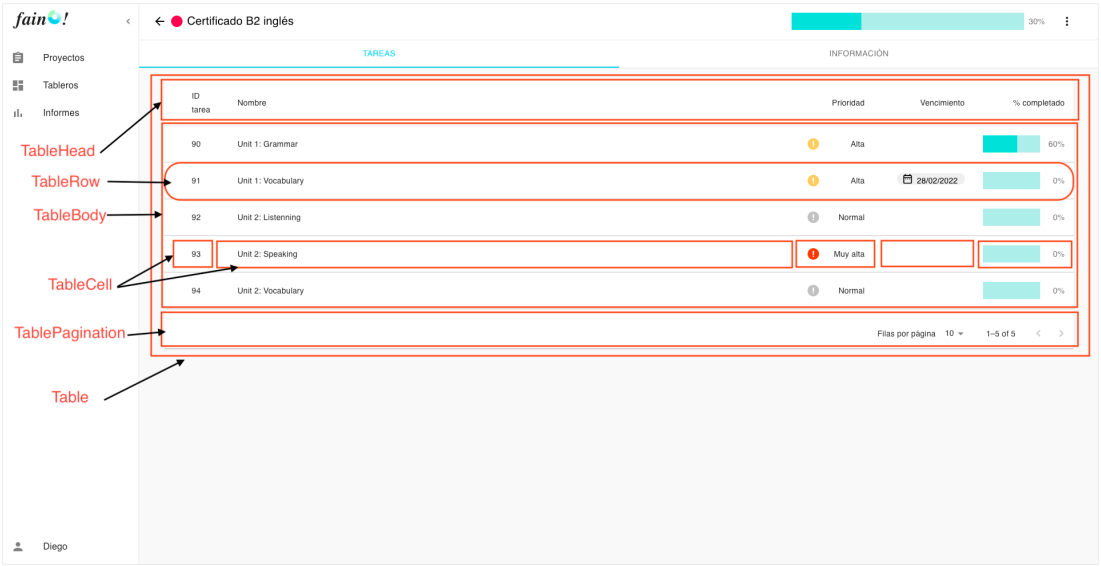

Figura 7.9: Vista da pestana de tarefas dun proxecto

A segunda pestana, correspondente á información dun proxecto, mostra o compoñente ***ProjectInfoTab***, o cal mostra un grid que por agora só terá un elemento no que se mostra a descrición e as datas de inicio e fin do proxecto. Mediante o uso do grid, o compoñente xa queda preparado para ser extendido no futuro con novos elementos, que corresponderán ás gráficas do proxecto. A estrutura deste grid é moi parecida ao exposto na lista de proxectos, tamén usa o compoñente ***Grid*** de MUI.

No CU-31 móstrase na figura 7.30 o contido desta pestana, cos informes incluidos. Na figura 7.10 móstrase o diagrama da xerarquía de compoñentes da páxina de detalle de proxecto, incluindo as dúas pestanas.

CU-07 Actualizar proxecto, CU-08 Eliminar proxecto

Estes dous casos de uso implementáronse simultáneamente. En primeiro lugar, no backend engadiuse unha operación PUT para actualizar un proxecto e unha DELETE para eliminalo.

No frontend engadiuse na cabeceira da páxina ***ProjectDetails*** un menú desple-

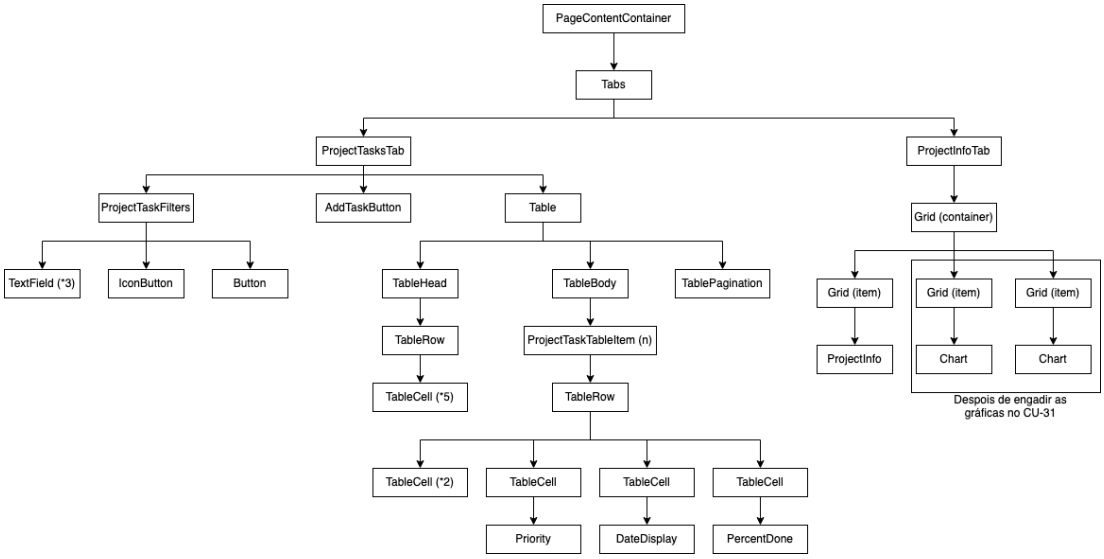

Figura 7.10: Diagrama de compoñentes do contido da páxina de detalle de proxecto

gable que contén as opcións de editar e eliminar (figura 7.11). Este menú utiliza o compoñente propio ***Menu***, o cal internamente usa o compoñente ***Menu*** de MUI, sobre o que realizan algunhas adaptacións.

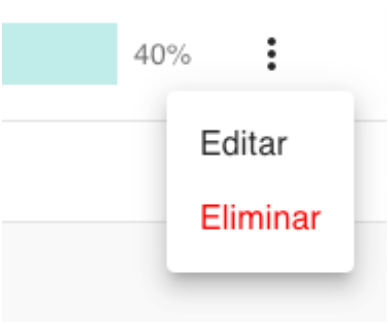

Figura 7.11: Desplegable na cabeceira da vista de detalle de proxecto

Premendo na opción de Editar móstrase a ventá modal do compoñente ***UpdateProjectModal***. que internamente reusa o compoñente anteriormente mencionado ***ProjectFormModal***, coa diferenza de que neste caso mostra os datos actuais cargados, para que o usuario os poida actualizar. (visto na figura 7.7)

Se se preme na opción de Eliminar abrirase unha ventá modal de confirmación, para así evitar eliminar por erro. Esta móstrase mediante o compoñente propio e reusable ***Confirmation***, que contén unha mensaxe de confirmación e as opcións de Aceptar ou Cancelar. Neste caso, se o usuario acepta, o proxecto e tódalas súas tarefas serán eliminados. Unha vez completada a acción, configúrase un callback para rediri-

xir ao usuario á lista de proxectos. Na figura 7.12 vese o compoñente ***Confirmation*** para este caso.

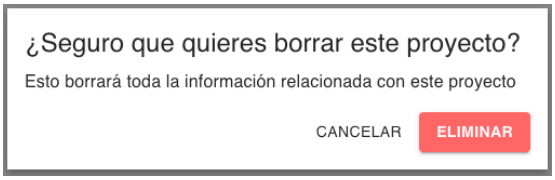

Figura 7.12: Vista da ventá modal de confirmación

CU-10 Crear tarefa proxecto

Nun inicio pensárase este caso de uso de forma que á hora de crear unha tarefa xa se pediran boa parte dos datos desta (nome, descrición, data límite, prioridade...). Pero finalmente dedidiuse que o único dato que se pediría neste paso sería o nome da tarefa. Esta decisión básase en que o feito de crear tarefas vai ser algo habitual para os usuarios, pero non tódalas tarefas precisan tódolos datos, pode haber unha boa parte delas que simplemente co nome sirva para definila. Polo tanto, considerouse mellor opción evitar pedir datos que é posible que non se necesitaran á hora de creala, centrando os esforzos en proporcionar unha experiencia áxil e agradable á hora de actualizar estes datos despois de crear a tarefa.

Polo tanto, no frontend só se necesitou engadir un botón na vista da lista de tarefas do proxecto, que unha vez pulsado habilite unha ventá modal cun formulario que únicamente pide o nome da tarefa. Ao confirmar dispárase a acción correspondente de Redux, que invoca a operación POST creada no backend. Cando esta se completa a modal cerrarase e refrescarase a lista de tarefas.

Figura 7.13: Ventá modal para engadir tarefa a proxecto

CU-09 Filtrar tarefas proxecto

Para implementar este caso de uso comezouse por modificar a operación do backend de listado de tarefas paxinadas permitindo que esta reciba uns filtros, que se

engadirán á consulta á base de datos permitindo devolver a lista de tareas filtrada e paxinada. Os filtros permitidos serán por palabras chave, por prioridade e por estado da tarefa (Rematada, Sen rematar). A maiores hai un filtro especial que permite dedidir se a lista de tarefas contén tamén as subtarefas no mesmo nivel, sen xerarquía, ou se pola contra nesa lista só veñen as tarefas raíz, xa que as subtarefas virán contidas nelas, permitindo despois na interface mostralas mediante un desplegable.

No frontend creouse un compoñente ***ProjectTasksFilters***, que se sitúa arriba da tabla de tarefas e está composto por tres ***TextField*** correspondentes a cada un dos filtros (dous deles de tipo select). A continuación dos campos de texto ten un icono que mostra cómo se está a mostrar a lista segundo o filtro especial mencionado, e no que se pode pinchar para mostrala da outra forma. Por último, este compoñente ten un botón para filtrar.

Cando se preme o botón de filtrar dispárase a acción correspondente en Redux. Esta acción actualiza o estado previamente creado para a lista de tarefas, polo que desta forma a tabla se refresca automaticamente cos resultados da búsqueda.

Na figura 7.14 móstrase a vista da lista de tarefas dun proxecto, cos filtros e o botón de crear tarefa engadidos.

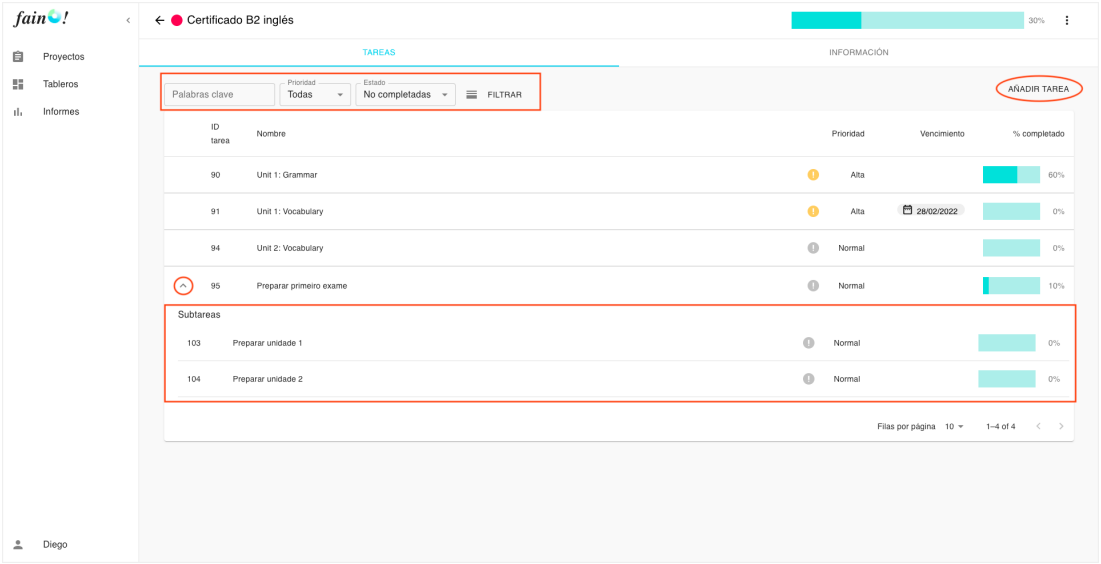

Figura 7.14: Vista da pestana de tarefas dun proxecto, mostrando filtros e desplegable de subtarefas

CU-11 Ver detalle tarefa, CU-12 Actualizar detalles tarefa, CU-13 Eliminar tarefa

Estes tres casos de uso realizáronse simultamneamente debido principalmente a

que na parte da interfaz íase traballar sobre a mesma parte, unha ventá modal que amosará os detalles da tarefa pero que servirá tamén para actualizar cada un destes por separado e dunha forma áxil e intuitiva.

En consecuencia, o primeiro que se fixo foi implementar as operacións necesarias no backend. Tratouse dun GET para devolver os detalles da tarefa, un DELETE para eliminar esta e un PATCH para actualizala. Como aclaración, para actualizar unha tarefa emprégase o PATCH en lugar de PUT debido a que, como se mencionou, preténdese poder actualizar cada valor da tarefa por separado, algo para o que está indicado o PATCH, xa que o PUT está indicado para recibir o recurso completo e remplazalo enteiro.

Pola parte do frontend creouse unha ventá modal que serviría para ver os detalles dunha tarefa. Esta é global na aplicación, xa que se poderá acceder a ela pinchando en calquera das tarefas da lista nun proxecto, e na seguinte iteración tamén dende as tarefas nun taboleiro. Para conseguir isto, creouse un hook que, recibindo un ID dunha tarefa, fai a chamada correspondente ó backend e posteriormente abre a ventá modal coa información desa tarefa. Como este hook se pode usar en calquera compoñente, xa se fai posible o feito de abrir esta ventá modal dende calquera parte da interface.

A ventá modal de detalle dunha tarefa está diferenciada en dúas partes, un contido e unha barra lateral, das que se mencionarán os detalles máis relevantes a continuación.

O compoñentes máis relevantes da barra lateral son:

- **SidebarActions:** Conxunto de iconos que disparan accións ao facer click neles. Atópase un icono para cerrar a modal da tarefa, e tamén outro para eliminala. Ao pinchar no de eliminar mostrarase a confirmación, como no caso de eliminar un proxecto.

- **TaskIDForm:** Trátase dun formulario composto por un único campo numérico que ten dous usos. En primeiro lugar serve para mostrar o ID da tarefa. Por outra parte, o usuario pode editar ese valor e premer na lupa, o que dispara a acción de buscar tarefa e posteriormente actualiza a modal cos datos da tarefa buscada.

- **InfoItem:** Compoñente xenérico creado para mostrar o resto da información da barra lateral. Por unha parte, permite mostrar un icono, un título e un valor principais. Tamén permite ser editable, polo que neste caso permite ser clickable, e recibe por prop un compoñente que servirá para editar o valor (este pode ser distinto para cada caso, razón de recibilo polas props). Este compoñente

mostrarase ao pinchar no elemento. Na figura 7.15 vese un exemplo detallado do funcionamento do compoñente.

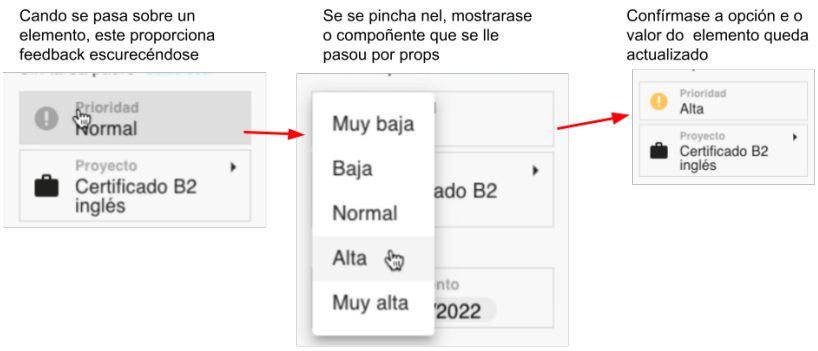

Figura 7.15: Funcionamento do compoñente xenérico InfoItem

Con respecto á parte do contido os compoñentes máis relevantes son:

- **TaskNameDisplay:** Este compoñente sirve tanto para mostrar como para actualizar o nome da tarefa. Desta forma, cando se abre a ventá modal este cárgase como un título, mostrando o nome da tarefa. No momento no que o usuario pincha neste título, o compoñente actualiza o seu estado interno, e pasa a estar "en edición", polo que en consecuencia renderiza un ***TextField*** co nome actual da tarefa cargado. O usuario poderá editalo e cando presiona enter o nome será actualizado e o compoñente volverá ao seu estado orixinal, sin estar en edición.
- **TaskDescriptionDisplay:** O seu funcionamento é como o do anterior compoñente. O único que varía é que neste caso a acción de permitir a edición ten lugar presionando un botón en vez de o propio texto da descrición.
- **PercentDone:** Este compoñente reusable xa estaba creado de outro caso de uso, pero foi necesario facerlle unha modificación para permitir a súa edición. Esta modificación consistiu en engadirlle unha prop "onSave", polo que se esta se lle pasaba ao compoñente, este permitiría ser clickable. Pinchando nel, mostra un desplegable coas posibles opcións para actualizar a porcentaxe, e cando se fai click nunha delas invoca o seu método onSave. Neste caso a implementación dese método o que fai é actualizar a porcentaxe de completitude da tarefa.
- **TaskCheckList:** Encárgase de mostrar a checklist dunha tarefa. Ten unha cabeceira co nome da sección e cun botón que permite engadir un novo elemento á checklist. Por outra parte, para mostrar os elementos utilízase o compoñente

List de MUI, mostrando cada elemento mediante un ***ListItem***. Este último está composto por un compoñente ***Checkbox*** que permite marcar ou desmarcar o elemento, outro ***ItemValue*** para mostrar o valor dese elemento e un último ***IconButton*** para permitir eliminar o elemento. Como aclaración, o compoñente ***ItemValue*** ten un comportamento idéntico a ***TaskNameDisplay*** para permitir ser editado.

- **Subtasks:** Este compoñente mostra as subtarefas da tarefa. Ten unha cabeceira coma a do anterior. As subtarefas son mostradas mediante o compoñente ***Table*** de MUI, no que para mostrar cada unha delas se reutilizará o compoñente previamente creado ***TaskTableItem***.

Na figura 7.16 móstrase a vista desta ventá modal, resaltando algúns dos compoñentes anteriormente mencionados.

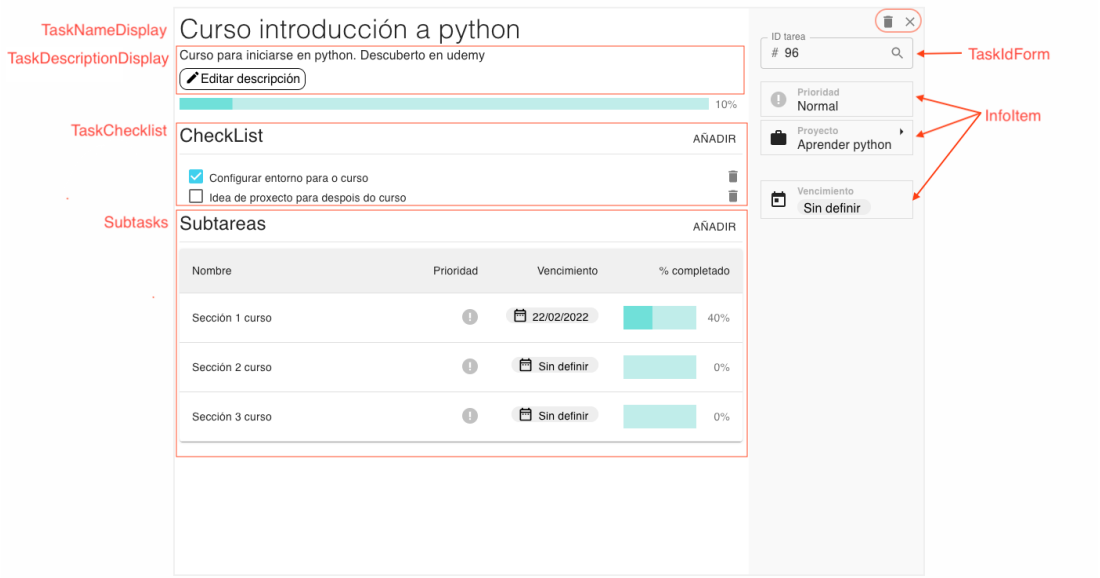

Figura 7.16: Ventá modal de detalle dunha tarefa

7.6 Iteración 4

7.6.1 Análise

O obxetivo desta iteración é implementar o relacionado coa sección de taboleiros. Desta forma, o resultado será unha aplicación que, a maiores da anterior iteración, permitiralle a un usuario crear taboleiros asociados a un intervalo de tempo, e poder

planificar as tarefas dos diferentes proxectos neses taboleiros. Así mesmo, poderá comprobar os seus avances ao ir movendo as tarefas sobre as diferentes columnas que dispoña no taboleiro, observando dunha forma visual o estado de cada tarefa. Os casos de uso correspondentes son:

- CU-14 Ver lista taboleiros
- CU-15 Crear taboleiro
- CU-16 Ver detalle taboleiro
- CU-17 Actualizar taboleiro
- CU-18 Eliminar taboleiro
- CU-19 Engadir columna a taboleiro
- CU-20 Actualizar nome columna taboleiro
- CU-21 Eliminar columna taboleiro
- CU-22 Engadir tarefa a taboleiro
- CU-23 Eliminar tarefa de taboleiro
- CU-24 Mover tarefa a outra columna dun taboleiro
- CU-25 Reordear tarefa en columna taboleiro

7.6.2 Deseño e implementación

CU-14 Ver lista taboleiros, CU-15 Crear taboleiro

A creación dun taboleiro implementouse dunha forma moi similar á de proxecto, pero neste caso xa se implementou xunto co caso de uso de listar taboleiros.

Comezouse implementando unha operación POST no backend, que recibe os datos do taboleiro e o crea se o usuario non ten outro co mesmo nome ou cun rango de tempo que colisione co que se vai a crear. Tamén se implementou a operación GET para recibir a lista de taboleiros paxinada, implementada tamén mediante un DAO propio ***BoardPaginationDaoImpl***.

No frontend, creouse un compoñente ***Boards*** usando o compoñente ***Page*** e se lle engadiu na cabeceira un botón para permitir crear taboleiros. Para a creación do taboleiro a estrutura de compoñentes é similar á de proxecto, polo tanto crease o compoñente ***CreateBoardModal***, que internamente usa o compoñente reusable

BoardFormModal, que mostra unha ventá modal cun formulario composto por un ***TextField*** para o nome, e dous ***DesktopDatePicker*** para as datas de inicio e fin.

Na figura 7.17 móstrase a vista da ventá modal de crear taboleiro.

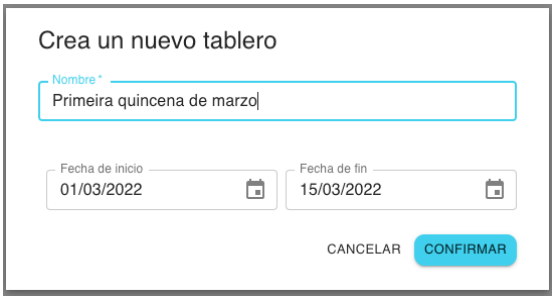

Figura 7.17: Vista da ventá modal para crear ou actualizar taboleiros

O contido da páxina representada polo compoñente ***Boards*** consiste nunha táboa cunha estrutura de compoñentes similar á da lista de tarefas dun proxecto, usando os compoñentes de MUI (***Table***, ***TableHead***, ***TableBody***, ***TableRow*** e ***TableCell***). Cada fila desta táboa ten tres celdas, unha para o nome do taboleiro e outras dúas para a data de inicio e fin, nas que se usa de novo ***DateDisplay***. Tamén se usa o compoñente ***TablePagination*** para manexar a paxinación.

Na figura 7.18 móstrase a vista da lista de taboleiros

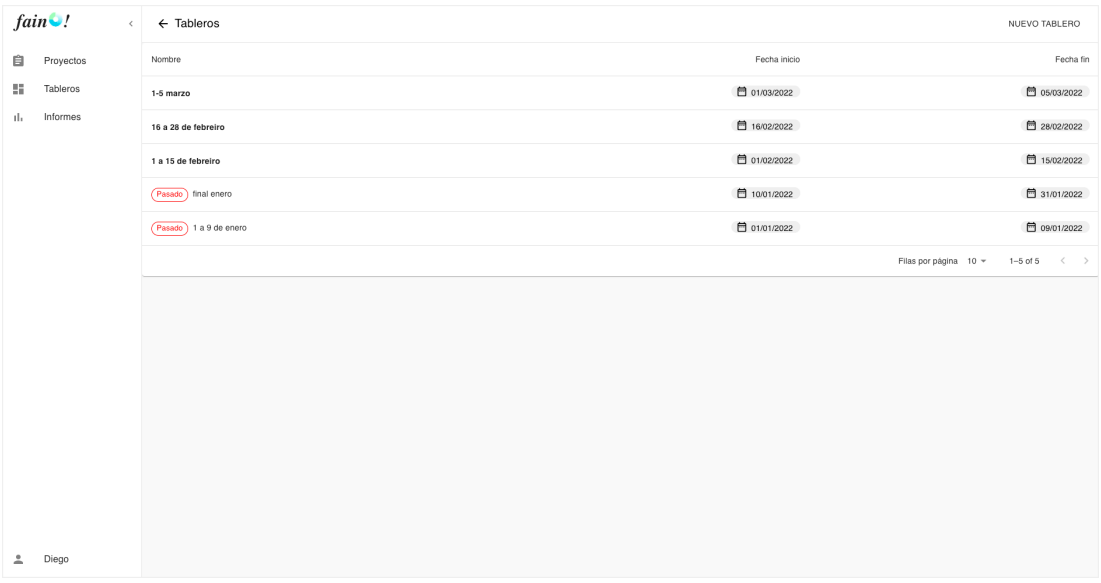

Figura 7.18: Vista da páxina de taboleiros

CU-16 Ver detalle taboleiro, CU-17 Actualizar taboleiro, CU-18 Eliminar taboleiro

Para a implementación destes casos de uso creáronse as operacións necesarias no backend: un GET por ID de taboleiro para devolver o detalle deste, un PUT para actualizar e un DELETE para eliminalo.

No frontend creouse o compoñente ***BoardDetails***, que representa a páxina de detalle dun taboleiro. Na cabeceira da páxina mostrouse o nome do taboleiro e as súas datas de inicio e fin, así como se habilitou, commo no caso dos proxectos, un menú coas opcións de Editar e Eliminar. Como se mencionou, neste momento o taboleiro non tiña nada máis polo que o detalle do taboleiro ata o momento atopábase todo na cabeceira, quedando o contido vacío.

Para actualizar os datos do taboleiro, o usuario preme no botón do desplegable e móstrase unha modal ***UpdateBoardModal***, que internamente usa ***BoardFormModal*** pero que xa mostra o formulario cos datos cargados. Por outra parte, para eliminar o taboleiro, ao premer na opción de eliminar reúsase o compoñente ***Confirmation*** para que o usuario confirme o borrado. Na figura 7.21 móstrase a estrutura base da páxina do taboleiro, despois da realización do caso de uso de engadir columnas. E na figura 7.19 móstrase as opcións da cabeceira da páxina de detalle dun taboleiro. Nesta xa se mostra a opción que se engadirá no seguinte caso de uso.

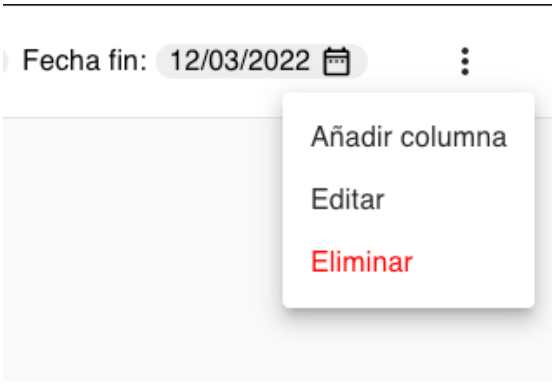

Figura 7.19: Opcións no desplegable da páxina de detalle dun taboleiro

CU-19 Engadir columna a taboleiro, CU-20 Actualizar nome columna taboleiro, CU-21 Eliminar columna taboleiro

Estes casos de uso foron realizados de forma simultánea xa que no backend as operacións eran rápidas de implementar, e no frontend o xeito de desenvolvelos á vez permitía aforrar esforzo.

No backend, por unha parte engadíronse as operacións necesarias, similares ás de

anteriores casos de uso. Por outra parte actualizouse a creación dun taboleiro para que se crean 3 columnas por defecto (ToDo, Doing, Done) para axudar ao usuario. Evidentemente, este poderá modificalas ou eliminalas se así o desexa. Adicionalmente tamén foi necesario actualizar a estrutura do DTO do taboleiro e o seu *converter* para incluír neste as súas columnas.

Para implementalo no frontend, en primeiro lugar engadiuse ao desplegable da páxina de detalle dun taboleiro unha opción adicional para engadir unha columna ao taboleiro. Esta opción abre unha ventá modal cun formulario cun único campo de texto para introducir o nome da columna. Ao confirmar chámase á operación do backend e o taboleiro queda refrescado no estado de Redux, polo que xa se actualizará a interface coa columna engadida.

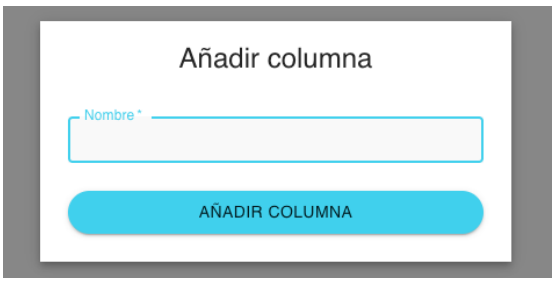

Figura 7.20: Ventá modal para engadir columna a un taboleiro

A continuación creouse un compoñente ***BoardContent***, que se usará como contido da páxina de detalle do taboleiro, e que é necesario para aplicar unha serie de regras de estilo para o contido do taboleiro. A continuación creouse ***BoardColumn***, o compoñente que renderizará cada columna do taboleiro. Este está composto por un ***ColumnHeader*** e a continuación o contido da columna.

O compoñente ***ColumnHeader*** serve para mostrar o nome da columna, permitindo tamén editalo. Polo tanto, cando se preme no nome desta habilítase o campo de texto e cando se preme enter dispárase a acción de Redux que chama ao método implementado no backend. Por outra parte, este compoñente de cabeceira tamén ten un icono dunha papeleira que permite eliminar a columna, mostrando antes a modal de confirmación.

Na figura 7.21 móstrase a estrutura base da páxina do taboleiro, que xa contén columnas, mostrando ademais a edición do nome dunha delas.

CU-22 Engadir tarefa a taboleiro, CU-23 Eliminar tarefa de taboleiro

Na parte do backend creáronse dúas operacións POST, unha para engadir unha tarefa a un taboleiro e a outra para eliminala deste. Estas reciben o ID da tarefa e o da columna á que se desexa engadir.

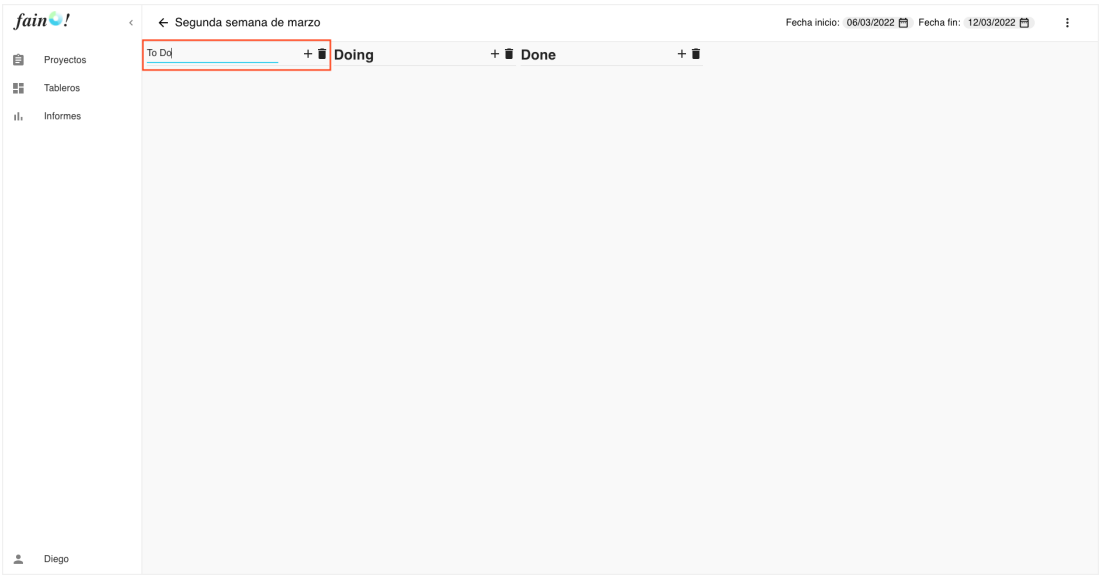

Figura 7.21: Vista base dun taboleiro vacío, mostrando a edición de columna

Unha tarefa pódese engadir a un taboleiro dende dous sitios diferentes. En primeiro lugar, na ventá modal de detalle de tarefa, engadíronse na barra lateral dous novos elementos, para taboleiro e columna, reusando o compoñente ***InfoItem***. Polo tanto, estes elementos mostrarán o taboleiro e columna no que se atopa a tarefa, permitindo modificar ambos dende ahí. Cando unha tarefa non está nun taboleiro, pódese engadir a un mediante o elemento do taboleiro. Na figura 7.22 explícase o proceso.

Por outra parte, unha tarefa tamén se pode engadir a un taboleiro, ou mellor dito a unha columna, dende a propia columna. Na cabeceira da columna, ao lado do botón da papeleira engadiuse un icono "+", que abre unha ventá modal ***AddTaskToBoardModal***. Aquí, en primeiro lugar débese seleccionar o proxecto no cal se atopa a tarefa, mediante o primeiro ***TextField*** da modal (que é de modo select). Unha vez seleccionado, cargaranse e mostraranse as tarefas dese proxecto que non están nese taboleiro. Para mostralas empregáronse os compoñentes ***List*** e ***ListItem*** de MUI. Pero tamén se mostra un segundo ***TextField***, que permite insertar o nome dunha tarefa, ben sexa para filtrar na lista ou para crear unha tarefa nova nese proxecto. Esta opción de crear unha tarefa dende aquí engadiuse para favorecer a experiencia do usuario e evitar que para iso tivera que ir á páxina de detalle do proxecto. Na figura 7.23 móstrase a vista desta ventá modal.

A continuación creouse o compoñente ***TaskBoardItem***, que mostrará a información de cada tarefa no taboleiro. Internamente este compoñente reutiliza outros xa

Figura 7.22: Engadir tarefa a taboleiro dende a ventá modal de detalle da tarefa

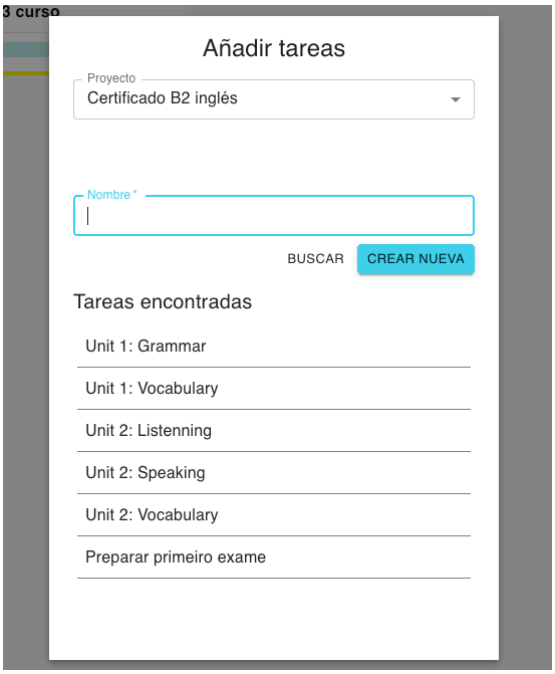

Figura 7.23: Ventá modal para engadir tarefas dende o taboleiro

creados previamente para mostrar a información (***Priority***, ***PercentDone***, ***DateDisplay***). O compoñente amén mostra o nome e ID da tarefa, así como unha liña coa cor do proxecto no que se atopa a tarefa, desta forma o usuario poderá identificar dun vistazo as tarefas dun proxecto concreto en función da cor. Por outra parte, tamén se

engadiu na esquina superior dereita deste un icono que cando se pincha nel desvincula esa tarefa do taboleiro. Na figura 7.24 amósase a vista deste compoñente para unha tarefa, e na 7.25 móstrase un taboleiro que contén tarefas de varios proxectos.

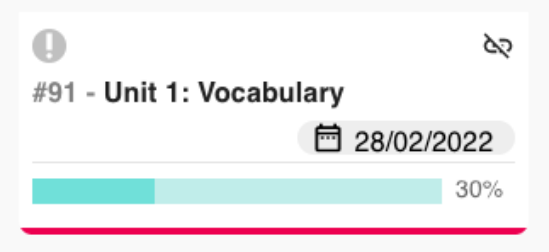

Figura 7.24: Vista do compoñente de tarefa dun taboleiro

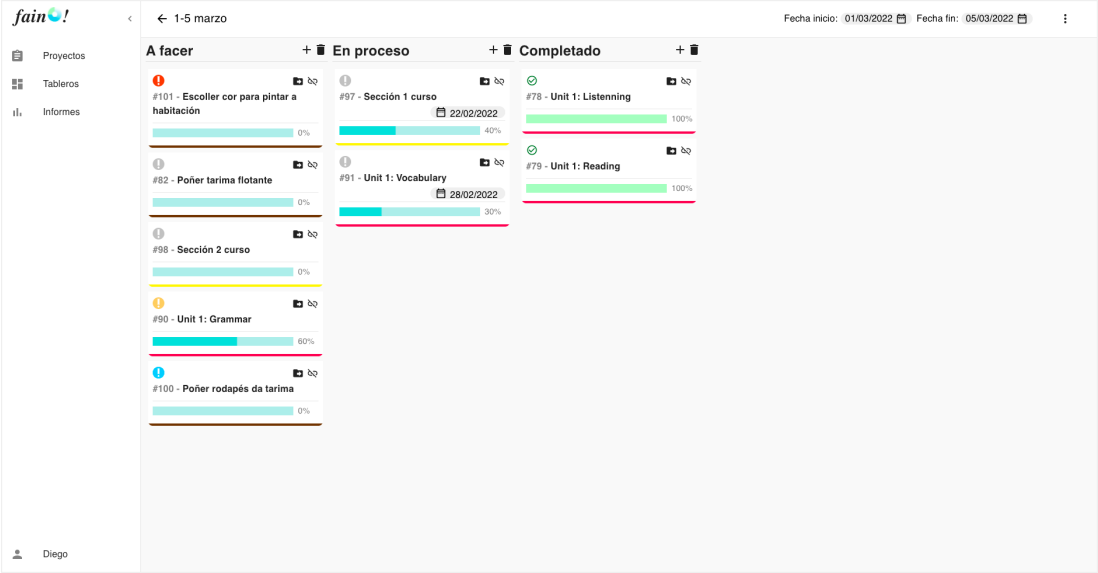

Figura 7.25: Vista dun taboleiro con tarefas de diferentes proxectos

CU-24 Mover tarefa a outra columna dun taboleiro, CU-25 Reordear tarefa en columna taboleiro

Unha boa experiencia do usuario con respecto a estes casos de uso era algo crítico para este proxecto, xa que non era aceptable que non fora mediante a forma intuitiva coa que se fai en calquera outra aplicación, é dicir, simplemente movendo a tarefa mentres se mantén pulsada co rato. Para isto era necesario o Drag and Drop (DnD), e para conseguilo usouse a librería react-beautiful-dnd [25]. Esta librería proporciona un maior nivel de abstración para implementar o DnD, e xenera este de forma moi eficiente, con movementos naturais e limpos, contando tamén con funcionalidades para garantir a accesibilidade (a11y). Esta librería é bastante sinxela de usar, pois con

un simple exemplo dos que teñen xa me serviu para ver cómo debería aplicalo ao meu caso.

En primeiro lugar, foi necesario crear un contexto para o DnD que ”envolvera” a tódolos compoñentes que se ían mover (as tarefas) ou que ían facer de contedores deles (as columnas), polo tanto o sitio adecuado para este contexto foi o compoñente ***BoardContent***, no cal se usou o compoñente ***DragDropContext*** de react-beautiful-dnd. O seguinte paso foi usar no compoñente ***BoardColumn*** o compoñente ***Droppable*** da librería, para así indicar que ese compoñente faría de contedor no DnD. Este compoñente recibe como prop *droppableId*, polo que ahí se establecerá co ID da columna. Finalmente quedaba indicar os elementos que se ían poder mover entre os contedores, é dicir, as tarefas, polo que no compoñente ***TaskBoardItem*** se usou o compoñente ***Draggable***. Este compoñente recibe por props un *draggableId* (que corresponderá ao ID da tarefa) e un *index* (correspondendo coa posición desa tarefa na columna).

Con isto xa se poderían mover as tarefas entre as diferentes columnas, pero aínda non hai forma de actualizar o estado, polo que ao remate de cada movemento a tarefa volvería ao seu lugar de orixe. Esta librería proporciona unha forma bastante sinxela de implementalo, xa que o compoñente ***DragDropContext*** recibe como prop unha función *onDragEnd*, que deberá ser implementada e proporciona o necesario para manexar o estado. Como o seu nome indica, esta función executarase cada vez que un movemento remate, e recibe por parámetro un obxeto result que contén o necesario para o manexo do estado:

- **source:** É un obxeto con dúas propiedades: *droppableId* e *index*. Polo tanto, o *droppableId* corresponderá co ID da columna de orixe e o *index* co índice que tiña a tarefa nesa columna.

- **destination:** Contén o mesmo que o anterior, pero neste caso con respecto á columna de destino.

- **draggableId:** Corresponde ao ID da tarefa que realizou o movemento.

Polo tanto, con esta información xa se pode invocar o método do backend correspondente en función dos valores destes parámetros para manexar tanto os movementos de tarefas entre columnas como o feito de reordear unha tarefa nunha columna, quedando así totalmente funcionalis os casos de uso de mover tarefas entre columnas e reordear tarefas na mesma columna.

7.7 Iteración 5

7.7.1 Análise

O obxetivo desta iteración é implementar o manexo do tempo dedicado ás tarefas, e posteriormente a sección de informes da aplicación. Desta forma, o resultado será unha aplicación que, a maiores da anterior iteración, permitiralle ao usuario ir engadindo rexistros de tempo a medida que vai realizando as tarefas, ben sexa de forma manual ou mediante o uso do cronómetro. Mediante a sección de informes, o usuario poderá ver gráficas mostrando dunha forma visual o seu histórico de tempo dedicado. Os casos de uso a abordar son os seguintes:

- CU-26 Iniciar cronómetro nunha tarefa
- CU-27 Parar cronómetro
- CU-28 Crear rexistro de tempo
- CU-29 Eliminar rexistro de tempo
- CU-30 Ver informes globais
- CU-31 Ver informes dun proxecto
- CU-32 Filtrar informe por data

7.7.2 Deseño e implementación

CU-28 Crear rexistro de tempo, CU-29 Eliminar rexistro de tempo

No backend creouse a operación POST para crear un rexistro de tempo, a cal recibe a duración en milisegundos e adicionalmente un comentario. Creouse tamén a operación DELETE para eliminalo. Por outra parte, foi necesario modificar a estrutura de DTO do detalle de tarefa e o seu convertidor para permitir devolver para unha tarefa tamén os seus rexistros de tempo.

No frontend, en primeiro lugar engadiuse un compoñente ***TaskTimeEntries*** que representa unha nova sección da modal da vista de tarefa polo que se usou o compoñente ***TaskContentSection***. Ao igual que as demais seccións, esta tamén ten un botón que permite engadir un novo rexistro de tempo. Polo que ao pulsalo se abre unha modal que contén un formulario con tres campos, que son ***DesktopDatePicker*** para a data do rexistro, que por defecto terá o día actual, e dous ***TextField***, un para

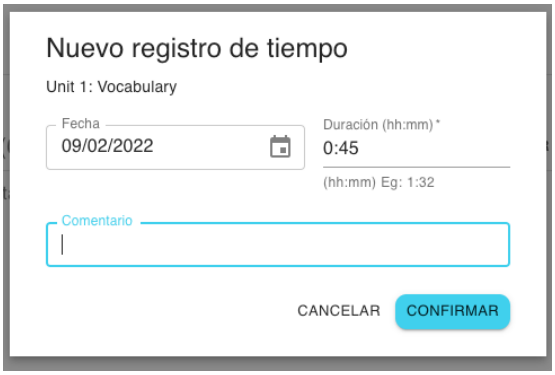

Figura 7.26: Vista da ventá modal para engadir un rexistro de tempo a unha tarefa

a duración e outro para o comentario. Na figura 7.26 móstrase a ventá modal para engadir un rexistro de tempo a unha tarefa.

Por outra parte, o contido desta nova sección corresponde ás entradas de tempo dunha tarefa, as cales se mostran empregando os compoñentes ***Table***, ***TableRow*** e ***TableCell*** de MUI. Cada fila da tabla está composta por catro celdas. A primeria mostra un icono dunha papeleira, que se usará para eliminar o rexistro de tempo. A segunda mostra a data na que se engadiu ese rexistro, empleando o compoñente ***DateDisplay***. A terceira corresponde ao comentario deste, quedando vacía no caso de que non teña comentario. Por último, a cuarta mostra a duración que se rexistrou, mediante o compoñente ***WorkedHours***.

Con este caso de uso xa se permitía o rexistro de tempo, polo que se mostrou esta información noutras partes da interface, tanto con respecto a tarefas como a proxectos. Na seguinte figura 7.27 móstrase a nova sección anteriormente mencionada así como os outros lugares donde se engadiu información sobre o tempo rexistrado.

CU-26 Iniciar cronómetro nunha tarefa, CU-27 Parar cronómetro

A información dun cronómetro era necesario persistila na base de datos, debido a que un usuario podería arrancar un cronómetro, pechar sesión na aplicación e volver entrar máis tarde paralo. Polo tanto foi necesaria unha entidade ***TrackerInfo***, que almacenaría o momento de rexistro do cronómetro xunto coa tarefa na que se empezou. Cabe mencionar que un usuario só poderá ter un cronómetro activo á vez.

En consecuencia, para estes casos de uso creáronse tres operacións no backend: unha para arrancar o cronómetro (persistir información), unha segunda para paralo (eliminar información persistida e devolvela ao cliente) e unha terceira para recuperar a información do cronómetro activo. Como aclaración, a operación de deter o cronómetro devolve a súa información no lugar de engadir o rexistro de tempo porque, como se verá a continuación, o usuario poderá introducir un comentario para

Figura 7.27: Información sobre o tempo empregado en tarefas e proxectos

ese rexistro de tempo.

No frontend, en primeiro lugar engadiuse un icono de Play na ventá modal de detalle de tarefa, arriba á dereita, para permitir iniciar o cronómetro nunha tarefa. A continuación creouse un compoñente ***TrackerInfo***, que se mostrará na barra lateral esquerda e servirá para ver o tempo que se leva cronometrado, en que tarefa e mostrará un icono para paralo. Na figura 7.28 vese o icono engadido e a vista do novo compoñente.

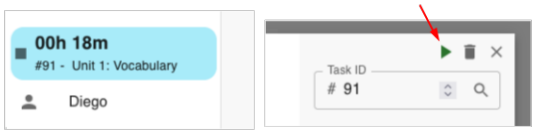

Figura 7.28: Cronómetro activo e botón de Play na vista de detalle de tarefa

Para manter actualizado o tempo mostrado no compoñente usouse un *setInterval*, que cada minuto actualiza o que ten renderizado para engadirlle un minuto máis. Cando se detén o cronómetro, non se rexistra automaticamente na tarefa, se non que se abre a ventá modal de engadir rexistro de tempo (figura 7.26) pero esta ten cargado o tempo do cronómetro que se acaba de parar. Polo tanto o usuario poderá rexistrar ese tempo, pero se quere poderalle engadir un comentario.

CU-30 Ver informes globais, CU-32 Filtrar informe por data

Como os informes proporcionarán información acerca do tempo empregado nas tarefas ou proxectos, no backend creouse unha implementación propia para extender o repositorio ***TimeEntryDao***. Esta implementación expón os métodos necesarios para a xeración dos informes, os cales se implementan definindo a consulta que se realizará á base de datos, e convirten os resultados nos obxetos correspondentes para devolver ao frontend.

No frontend empregouse a librería Apache ECharts [26]. Na súa páxina proporciona gran cantidade de exemplos de tódolos tipos de gráficas que soporta, polo que foi sinxelo atopar os que se adaptaban ao meu caso. A forma de xerar gráficas con esta librería é sinxela, xa que só é necesario o uso dun compoñente (***ReactECharts***) ao cal se lle pasa por props un obxeto que contén a configuración necesaria para mostrar a gráfica correspondente. A continuación explícase o proceso seguido para xerar cada un dos informes globais:

IG-01 Traballo total diario: Este foi o primeiro informe da aplicación. A súa función é a de mostrar unha gráfica circular na que se detallen as tarefas ás que se lle dedicou tempo no día actual e o tempo dedicado a cada unha. Polo tanto, estará dividida en seccións, correspondendo cada unha a unha tarefa, e variando o tamaño en función do tempo dedicado. Para implementalo, no backend creouse unha consulta que busca a suma dos rexistros de tempo do día actual e os devolve agrupados por tarefa, devolvendo para cada tarefa o seu nome e a duración total do tempo adicado no día.

No frontend, en primeiro lugar creouse un compoñente ***Chart***, que encapsula ao compoñente ***ReactECharts***, engadíndolle algo de lóxica reusable entre tódalas gráficas do frontend. A continuación creouse o compoñente ***SpentTimeOnTasksToday***, que obtén os datos para xerar a gráfica, mediante a acción correspondente de Redux, e despois xera o obxeto de configuración da gráfica, que lle pasará ao compoñente ***Chart***. Un exemplo do obxeto de configuración xerado para esta gráfica circular é algo así:

```
{
    series: [
      {
        type: "pie",
        radius: ["55%", "90%"],
        data: [
            { value: 1.23, name: 'Tarefa 1' },
            { value: 1.1, name: 'Tarefa 2' },
            { value: 0.6, name: 'Tarefa 3' },
```

```
            { value: 0.43, name: 'Tarefa 4' },
            { value: 0.25, name: 'Tarefa 5' }
          ]
        },
      ],
};
```

A propiedade máis importante deste obxeto é *series*, na que se declara o tipo de gráfica e os seus datos, que como se pode ver, teñen a mesma estrutura que os devoltos polo backend. Este obxeto permite moitas outras propiedades de configuración, sobre todo para personalizar a apariencia da gráfica e o feedback ao usuario na súa interacción con esta. Estas propiedades foron omitidas do exemplo por simplicidade.

A vista da gráfica xerada por este informe pódese ver na figura 7.29. Esta gráfica é a que se atopa arriba á esquerda.

IG-02 Traballo total por proxecto: Este informe debe mostrar unha gráfica circular na que se detalle o tempo total dedicado a cada un dos proxectos. É similar ao anterior informe, polo tanto na parte do backend a consulta devolve a suma das entradas de tempo para cada proxecto. Esta consulta a maiores permite recibir dúas datas para filtrar o informe para un intervalo de tempo.

No frontend creouse o compoñente ***TotalTimeByProject***, que é similar ao compoñente creado para o anterior caso de uso, recibindo os datos do backend e xerando un obxeto de configuración, para usalo no compoñente ***Chart***. Ao tratarse dunha gráfica circular, este obxecto é similar ao exposto no anterior informe. A maiores, no compoñente ***TotalTimeByProject*** hai dous ***DesktopDatePicker*** para poder filtrar a gráfica para certo período de tempo.

A vista da gráfica xerada por este informe pódese ver na figura 7.29. Esta gráfica é a que se atopa arriba á dereita.

IG-03 Detalle tempo de traballo por proxecto: Aquí mostrarase unha gráfica de barras na que cada barra corresponde a un día. Esa barra estará dividida en seccións, correspondendo cada unha a un proxecto e ao tempo dedicado a el nese día. Na parte do backend creouse a consulta que tamén permite filtrar por período e devolve, para cada proxecto ó que se lle dedicara tempo no período, o tempo total dedicado en cada un dos días dentro dese período.

No frontend creouse un novo compoñente ***DailySpentTimeByProject***, o cal contén os ***DesktopDatePicker*** para poder filtrar a gráfica para certo período de tempo, e usa o compoñente ***Chart*** para mostrala. O obxeto de configuración para unha gráfica de barras é semellante ao da circular. O que cambia é que neste caso ao representar varios proxectos en cada barra, hai que crear unha serie de datos de tipo *bar* para cada

proxecto a mostrar. E en cada serie irán os datos dese proxecto. Esta gráfica é a que aparece abaixo na figura 7.29.

Estes tres compoñentes correspondentes aos informes globais disporanse nunha nova páxina, representada polo compoñente ***Reports***. Esta terá dúas pestanas, unha para os informes globais e outra para os informes de proxecto, que se explicará no seguinte caso de uso. Para as pestanas reusouse o compoñente ***Tabs***, usado tamén na páxina de detalle de proxecto. O contido da pestana dos informes globais é un grid de tres elementos, un por cada un dos anteriores informes. Polo tanto úsase o compoñente ***Grid*** de MUI, tanto para o contedor como para os elementos.

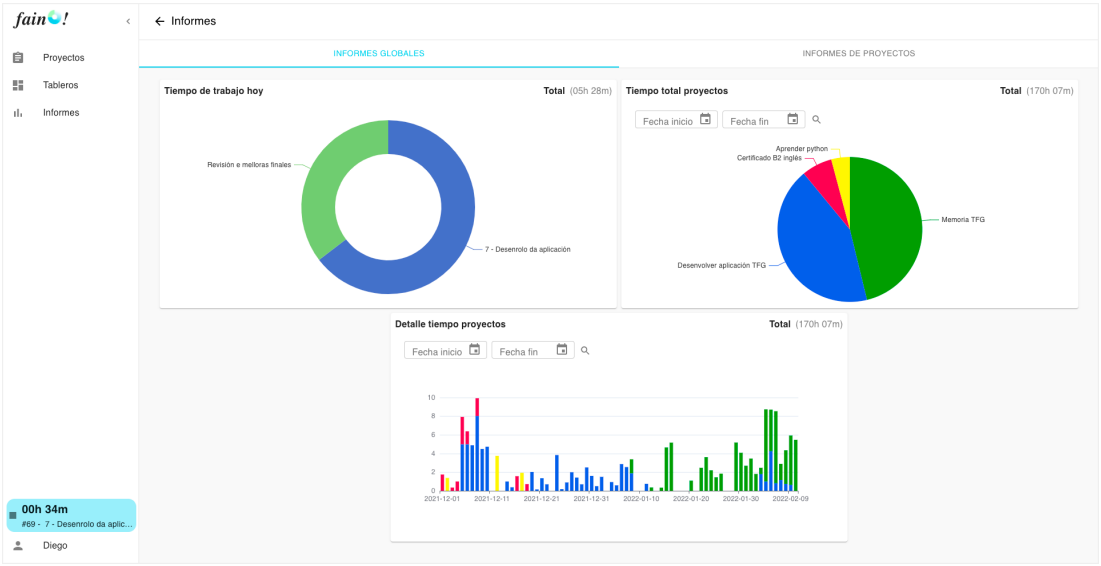

Figura 7.29: Vista da pestana de informes globais

CU-31 Ver informes dun proxecto, CU-32 Filtrar informe por data

IP-01 Estado das tarefas do proxecto: Este informe mostrará unha gráfica circular na que se detallará o número de tarefas dun proxecto que se encontran sen empezar, en proceso e rematadas. No backend este informe implementouse con métodos no DAO de tarefa, empregando a nomenclatura de JPA, sendo estes autoxenerados. Con iso xa serviu, xa que só era necesario consultar as tarefas en base á súa porcentaxe de completitude.

No frontend engadiuse un compoñente ***ProjectTasksCompletionStatus***, que ao igual que os demais, xera a gráfica circular correspondente. A vista desta gráfica atópase na figura 7.30, arriba á dereita.

IP-02 Detalle tempo de traballo por tarefa: Mostrará unha gráfica de barras na que cada barra corresponde a un día, e nesa barra se mostra unha sección por cada

tarefa á que se lle dedicara tempo ese día, correspondendo o tamaño da sección co tempo dedicado ese día a esa tarefa. Esta gráfica é como a explicada para o caso dos proxectos, pero aplicado ás tarefas dun proxecto. Na figura 7.30 esta é a gráfica que aparece abaixo.

Estes dous compoñentes engadíronse como novos elementos ao grid da pestana de información dun proxecto, representada polo compoñente ***ProjectInfoTab***. Polo tanto, desta forma estas gráficas xa se poden ver dende a páxina de detalle do proxecto. A continuación, para facelas accesibles tamén dende a páxina de informes (***Reports***) fíxose o seguinte. Creouse o compoñente para a pestana de informes de proxecto, ***ProjectReportsTab***, o cal en primeiro lugar mostra un ***TextField*** en modo select, para permitir seleccionar entre un dos proxectos do usuario. Unha vez seleccionado, móstrase o compoñente ***ProjectInfoTab*** para ese proxecto. Se volve a seleccionar outro proxecto diferente, a información refrescarase coa do proxecto seleccionado.

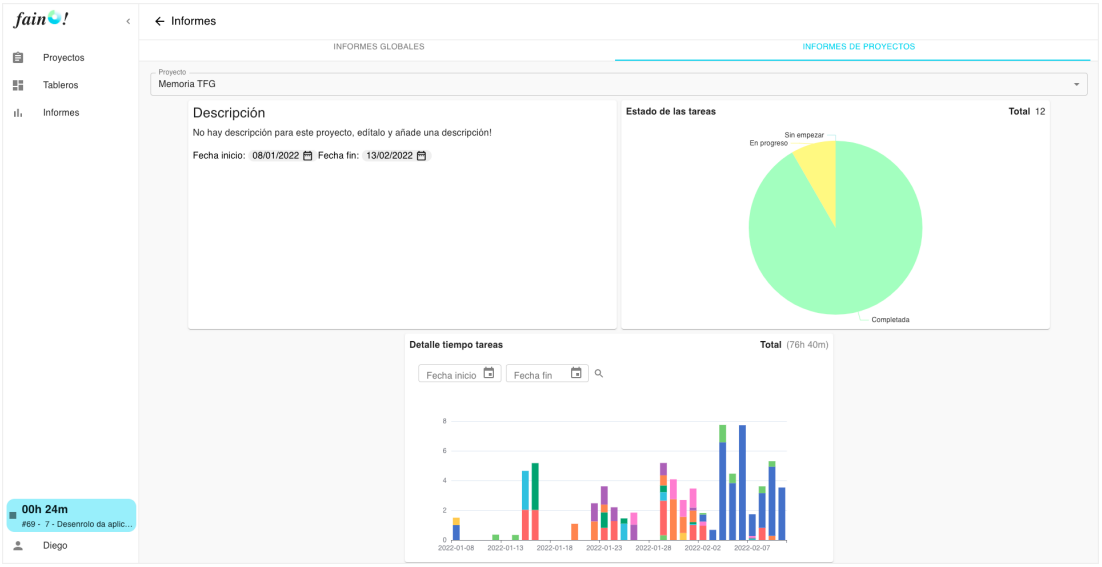

Figura 7.30: Vista da pestana de informes de proxecto

Estes xa serían tódolos informes planificados nos requisitos, pero neste momento considerouse interesante engadir un a maiores, a nivel de tarefa. Este mostra o tempo dedicado á tarefa nunha gráfica de barras, sendo cada barra un día. É semellante ás anteriores gráficas de barras, pero sen estar a barra dividida en seccións. Para mostralo engadiuse na ventá modal de detalle de tarefa, na sección de rexistros de tempo, un botón que permite cambiar a vista da tabla á nova gráfica. Na figura 7.31 móstrase esta vista.

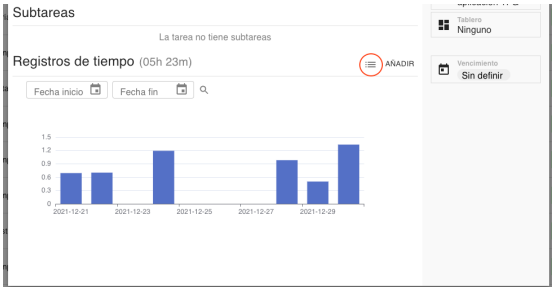

Figura 7.31: Vista da gráfica de tempo dunha tarefa por días

7.8 Iteración 6

A última iteración do proxecto dedícase á realización desta memoria. Como para a súa planificación xa se emprega a propia aplicación desenrolada, tamén se van detectando e implementando pequenas melloras na aplicación, sobre todo na interface de usuario. Algunhas das melloras realizadas son:

- Ordear os taboleiros de mais recente a menos, simplemente modificando a consulta á base de datos, e por outra parte mostrar unha etiqueta indicativa cando a data de fin do taboleiro xa pasou.
- Engadir a posibilidade de filtrar por rango de tempo na gráfica circular do tempo total dedicado por proxectos, xa que inicialmente nesta gráfica non era posible.
- Mostrar tódolos valores asociados á duración formateados en horas e minutos, xa que había algúns que se estaban a mostrar simplemente cun número decimal.
- Engadir feedback cando non se atopa unha páxina da app, cando a lista de proxectos, tarefas ou taboleiros non contén elementos, etc. Por exemplo, o que se fixo nestes casos foi mostrar un icono e un texto aclarativo, para evitar así mostrar unha páxina sen contido.
- Bastantes melloras de pequenos detalles de estilo para conseguir unha apariencia máis profesional.

Capítulo 8

Conclusións e traballo futuro

8.1 Conclusións

O principal obxectivo deste proxecto foi o de crear unha aplicación web que permita que unha persoa se organice adecuadamente e saiba exactamente en que emprega o seu tempo. Dado que tódalas iteracións planificadas foron completadas, pódese considerar que este obxectivo se cumpriu.

Por outra banda, tamén se buscaba unha aplicación que tivera unha interface de usuario sinxela, áxil e intuitiva para o usuario. Despois de realizar probas informais con algúns usuarios, tamén se pode considerar como satisfeito. A maiores, isto proporcionou unha aprendizaxe en novas librerías. Por unha parte creouse unha interface de usuario cun acabado profesional grazas aos compoñentes que ofrece MUI. Tamén se usou react-beautiful-dnd [25] para lograr o Drag and Drop das tarefas nas columnas do taboleiro. Finalmente, o uso de Apache ECharts [26], para mostrar gráficas cunha gran apariencia ao usuario.

Por último, este proxecto tamén serviu para consolidar os coñecementos adquiridos no grao, e máis en concreto, na mención de Enxeñaría do Software. Por unha parte, adquiriuse máis experiencia nas tecnoloxías usadas, xa coñecidas do grao, pero que se consideraba importante, xa que estas teñen unha alta demanda no mercado laboral. E por outra parte tamén se aplicaron tódolos coñecementos adquiridos á hora de analizar os requisitos e deseñar unha aplicación cunha arquitectura modular, con partes independentes e desacopladas e que favorecera a incorporación de novos cambios ou do mantemento desta.

8.2 Traballo futuro

Desde o primeiro momento, o produto concebiuse para que, se se desenrolaban tódalas funcionalidades plantexadas, a aplicación proporcionara un modelo completo de xestión de tarefas e tempo persoais, sinxela e sen funcionalidades que non foran imprescindibles, e que xeraran ruido para o usuario. Non obstante, hai algunhas funcionalidades que se considera interesante engadir no futuro.

A primeira é engadir a posibilidade de usar a técnica Pomodoro [4]. Desta forma, cando o usuario vaia dedicar tempo a unha tarefa, tería a posibilidade de facelo mediante un temporizador de conta atrás, de xeito que cando este acabara se podería rexistrar o tempo na tarefa, correspondente a un ”pomodoro”. Ao engadir este método tamén se poderían engadir novos informes para o usuario, como por exemplo o número de ”pomodoros” que realizou nun período de tempo, ou o número de ”pomodoros” nos que non chegou ao final do temporizador debido a unha interrupción.

Por outra parte, algo interesante para seguir mellorando a experiencia do usuario é engadir atallos de teclado. A lista de atallos dispoñibles sería accesible dende a aplicación. A continuación móstrase algúns exemplos de atallos útiles:

- Engadir algúns atallos para navegar axilmente entre as seccións da aplicación.
- Dende a vista da lista de proxectos ou taboleiros, atallos para abrir a ventá modal de crear proxecto ou taboleiro.
- Dende a vista de detalle dun proxecto, atallos para abrir a ventá modal de edición do proxecto e de creación de tarefa.
- Na ventá modal de detalle dunha tarefa engadir atallos para acceder á actualización de calquera dos seus campos. Por exemplo: actualizar a prioridade, o nome, engadir elemento para a checklist, etc.

Por último, sin ser unha nova funcionalidade, poderíase abordar a creación dunha aplicación móbil. Isto débese a que actualmente, se un usuario desexa levar o control de tarefas nas que non dispoña do ordenador, vaille resultar máis difícil o xeito de ter que recordar o tempo adicado para despois rexistralo, ou incluso ter que ir ao ordenador expresamente para iniciar o cronómetro. Cunha aplicación móbil, este problema quedaría resolto, proporcionandolle aos usuarios unha maior flexibilidade e comodidade á hora de levar o control de todo tipo de tarefas que desexen. Esta aplicación sería desenrolada usando o framework Ionic [27], polo que boa parte dos compoñentes da interface poderían ser reutilizados.

Apéndices

Relación de Acrónimos

API Application Programming Interface. 4, 30, 34

DAO Data Access Object. 29, 32, 37, 48, 61

DnD Drag and Drop. 54, 55, 65

DOM Document Object Model. 27, 30

DTO Data Transfer Object. 29, 51, 56

IDE Integrated Development Environment. 26

JRE Java Runtime Environment. 25

JSX JavaScript Syntax eXtension. 27

JWT JSON Web Token. 34, 36

POM Project Object Model. 25, 26

REST Representational State Transfer. 4, 29, 30, 32, 34, 37

SCM Software Configuration Management. 28

SPA Single Page Application. 4, 26

XML eXtensible Markup Langugae. 25, 26

Glosario

backend Parte do lado servidor dunha aplicación, encargada da lóxica de negocio e xestión de datos desta.. 4, 19, 29, 30, 32, 34–38, 40, 41, 43, 45, 48, 50, 51, 55–57, 59–61

framework Conxunto estandarizado de conceptos, prácticas e criterios para enfocar unha problemática particular.. 25, 28

frontend Parte do lado cliente dunha aplicación, que interactua directamente cos usuarios.. 4, 19, 30, 32–35, 38–41, 43–45, 48, 50, 51, 56, 58–61

hook É unha función que permite enganchar os compoñentes React funcionales a características propias dos compoñentes de clase, proporcionándolles un estado e ciclo de vida.. 30, 33, 45

Kanban Metodoloxía que se utiliza para visualizar un fluxo de traballo, que ten como elemento principal un taboleiro, que está dividido en columnas. Cada columna representa unha fase dese proceso. Cada tarefa que entra no fluxo entrará no taboleiro como unha tarxeta kanban, e irá pasando polas diferentes columnas en función da fase na que se atope.. 2, 5–8, 28

Pomodoro Técnica que consiste en usar un temporizador para dividir o traballo en bloques de tempo enfocado (xeralmente 25 minutos) separados por un breve descanso (xeralmente 5 minutos).. 7, 66

Bibliografía

[1] Páxina web de trello. [En liña]. Dispoñible en: https://trello.com/es

[2] Páxina web de todoist. [En liña]. Dispoñible en: https://todoist.com/es/

[3] Páxina web de kanbanflow. [En liña]. Dispoñible en: https://kanbanflow.com/features

[4] Explicación da técnica pomodoro. [En liña]. Dispoñible en: https://protecciondatos-lopd.com/empresas/tecnica-pomodoro/#Para_que_se_usa_esta_tecnica

[5] Páxina web de microsoft to do. [En liña]. Dispoñible en: https://todo.microsoft.com/tasks/es-es

[6] Páxina web de 135 list. [En liña]. Dispoñible en: https://135list.com/

[7] Páxina web de things. [En liña]. Dispoñible en: https://apps.apple.com/es/app/things-3/id904237743

[8] Páxina web de notion. [En liña]. Dispoñible en: https://www.notion.so

[9] Páxina web de 135 list. [En liña]. Dispoñible en: https://scrumguides.org/

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

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

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

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

[14] Páxina web de postman. [En liña]. Dispoñible en: https://www.postman.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 mui. [En liña]. Dispoñible en: https://mui.com/

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

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

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

[21] Páxina web de electron framework. [En liña]. Dispoñible en: https://www.electronjs.org/

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

[23] Páxina web de docker. [En liña]. Dispoñible en: https://www.docker.com/

[24] Introdución a json web token. [En liña]. Dispoñible en: https://jwt.io/introduction

[25] Librería react beautiful dnd. [En liña]. Dispoñible en: https://github.com/atlassian/react-beautiful-dnd

[26] Páxina web de apache echarts. [En liña]. Dispoñible en: https://echarts.apache.org/en/index.html

[27] Páxina web de ionic framework. [En liña]. Dispoñible en: https://ionicframework.com/