FACULTADE DE INFORMÁTICA

TRABALLO FIN DE MESTRADO

MESTRADO UNIVERSITARIO EN ENXEÑERÍA INFORMÁTICA

Deseño e implementación dun videoxogo no ámbito dos dereitos humanos baseado en ferramentas libres

Autor: Daniel José Barbeira Hayes

Directores: Sergio Fernández Alonso

Miguel Rodríguez-Segade Alonso

Emilio José Padrón González

A Coruña, Setembro 2018

If the users don't control the program, the program controls the users.

Richard M. Stallman

Resumo: O presente documento recolle a memoria do desenvolvemento dun videoxogo de tipo *tetris* denominado *"Tetris dos dereitos humanos"* no ámbito das tecnoloxías para o desenvolvemento humano (TpDH).

A idea clave é o concepto de ludificación (*gamification*), que consiste no uso de técnicas e mecánicas de xogos en contornos non lúdicos coa finalidade de potenciar a motivación e estimular a aprendizaxe. En concreto, neste videoxogo preténdese impulsar a aprendizaxe de novos conceptos e valores na educación no ámbito dos dereitos humanos engadindo contido sensibilizador.

O videoxogo terá unha apariencia sinxela utilizando as cores para diferencias os distintos servizos básicos relacionados cun dereito humano. En canto ás pezas, empregarase o conxunto de pezas clásico do *tetris* xunto con pezas personalizadas que poden ser de maior ou menor complexidade. Inclúense pequenas informacións en texto relacionadas con cada tipo de peza e haberá distintos niveis de dificultade. Cada nivel correspóndese cun país e a dificultade establécese en función do estado dos dereitos humanos en dito país. Cada dereito humano tratado conforma un subnivel distinto dentro do nivel do país.

O videoxogo será multiplataforma, podendo ser xogado tanto en computadoras persoais coma en dispositivos móbiles. Sendo este un requisito imposto, optouse polo uso de tecnoloxías web (HTML5, CSS3 e JavaScript) coa idea de aproveitar a capacidade de adaptación a dispositivos móbiles destas tecnoloxías.

Por último, é importante destacar o seguimento da filosofía e pautas de software libre en todo o proxecto. O código e a documentación serán publicados cunha licenza libre e optouse en todo momento pola utilización de estándares e ferramentas/tecnoloxías que cumpran con esta condición.

Lista de palabras chave:

Dereitos Humanos, Educación para o Desenvolvemento, Tecnoloxías para o Desenvolvemento Humano, Software Libre, Ludificación, Videoxogo, Tetris, JavaScript, HTML5, Scrum

Ferramentas e tecnoloxías empregadas:

- **Hardware:**
  - **Desenvolvemento e probas**: Portátil MSI GL62 6QF. Intel® Core i5-6300HQ (2.3 GHz, 6 MB cache), 8GB RAM DDR4, disco duro 1TB (7200 rpm S-ATA), tarxeta gráfica Nvidia® GeForce GTX 960M 2GB GDDR5.
  - **Probas e despregamento**: Servidor ESF.
- **Software:**
  - S.O. desenvolvemento e probas: GNU/Linux [...]
  - Linguaxes: HTML5, CSS3, JavaScript
  - Librerías: jQuery 3.1.1, SweetAlert2, jQuery.i18n, CLDRPluralRuleParser.js jQuery.history.js, JQVMap
  - Compilación: Browserify + Gulp
  - Outras ferramentas: GitLab, JsHint, http-server, Visual Studio Code

Índice xeral

Índice de figuras

Índice de cadros

Capítulo 1

Introdución

Este proxecto foi desenvolvido en colaboración con Enxeñaría Sen Fronteiras Galicia (ESF), unha asociación independente, sen ánimo de lucro e baseada no voluntariado. Dito proxecto enmárcase no ámbito da Educación para o Desenvolvemento (EpD) enfocado na difusión das *Tecnoloxías para o Desenvolvemento Humano*. A Tecnoloxía para o Desenvolvemento Humano (TpDH) consiste nun recoñecemento da necesidade de orientar o progreso tecnolóxico á promoción do desenvolvemento humano.

ESF pretende especializarse neste enfoque, e dende hai tempo quere explorar novos xeitos de achegar contidos sensibilizadores á poboación a través de canles populares. O xogo é unha destas canles, sendo importante para impulsar a aprendizaxe de novos conceptos a ludificación, unha técnica que aplica mecánicas de xogos en contornos non lúdicos coa finalidade de potenciar a motivación. Para este proxecto decidiuse desenvolver un ***tetris***. O videoxogo implementado pode ser usado tanto en PC coma en dispositivos móbiles, estando baseado en tecnoloxías web. Inclúe un motor co bucle do xogo que é utilizado coma un plugin de jQuery para poder crear a web que contén o xogo completo. ESF dispón dun servidor web no cal estará aloxado o xogo accesible públicamente.

1.1. Contexto e antecedentes

Dentro do plan estratéxico de ESF (2013-2017) inclúese coma un dos piares de actuación máis importante da asociación o traballo na EpD. Varias liñas estratéxicas de traballo previstas no plan entran nesa área de trabalo, sendo unha desas liñas a transversalización de contidos en dereitos humanos e EpD nos estudos universitarios tecnolóxicos cun enfoque na difusión das TpDH, onde ESF pretende especializarse. Ademais da elaboración de materiais didácticos e da organización de xornadas no eido formal e non formal, ESF quere explorar novos xeitos de levar eses contidos á poboación "non sensibilizada".

Desta premisa nace a idea de elaborar un videoxogo de contido sensibilizador no eido dos dereitos humanos. Deste xeito inténtase sensibilizar á xente de que pode xogar co produto (e pode ser que nin se fixe noutros contidos, de aí o desafío técnico e gráfico de visibilizar os contidos de dereitos humanos integrados coa dinámica do xogo), e ademais usar isto para transversalizar actividades cun enfoque de Educación para o Desenvolvemento.

1.2. Obxectivos

O principal obxectivo deste proxecto é levar a cabo o desenvolvemento completo do videoxogo *"Tetris dos Dereitos Humanos"*, que debe cumprir coas seguintes condicións:

- Poder ser xogado tanto en PC coma en dispositivos móbiles, independentemente dos seus sistemas operativos.
- Internacionalización dos textos en galego e castelán, facilitando ademais a adaptación dos textos a novas linguas.
- Dispor de xogos de pezas de distintas cores e formas, incluíndo as clásicas do *tetris* e novas pezas orixinais.

- Implementar unha forma de parametrizar o xogo que permita establecer a velocidade do xogo, a cor das pezas, criterios de finalización dun nivel, obstáculos adicionais, etc.
- Incluír pequenas informacións en texto xunto coa xeración das pezas a encaixar durante a partida.
- Estruturar o xogo nunha serie de niveis cos seus correspondentes subniveis.
- Rexistrar as puntuacións dos usuarios para controlar o seu progreso.

Adicionalmente, establécese tamén como obxectivo a obtención dun produto versátil e extensible, para o que se considera importante dispor, no posible, dun motor de videoxogo para un *tetris* o máis desacoplado posible do xogo *"Tetris dos Dereitos Humanos"* que se propón, de xeito que sexa doado tanto estender o videoxogo creado como crear outros xogos de tetris baseados noutras temáticas empregando o mesmo motor desenvolvido.

Ademais, todas as tecnoloxías implicadas no desenvolvemento do proxecto deben ser libres, de acordo coa definición de Software Libre da *Free Software Foundation* [1]. Todo o código e material extra desenvolvido licenciarase cunha licenza libre. O videoxogo desenvolvido integrarase nos sistemas de ESF e será distribuído a través das súas canles.

Como derradeiro obxectivo, propoñémonos rematar o proxecto con data límite o 5 de setembro de 2018.

1.3. Estrutura da memoria

Na presente memoria exponse o proceso de desenvolvemento do proxecto e explícanse a metodoloxía e os procedementos empregados para poder cumprir cos obxectivos marcados. Os capítulos repártense do seguinte modo:

Capítulo 1. Introdución

Neste capítulo descríbese brevemente a finalidade do proxecto e o porqué da necesidade da súa realización. Inclúese a motivación do mesmo, os obxectivos xerais e unha explicación da estrutura da memoria.

Capítulo 2. Xestión do proxecto

Inclúe a explicación da metodoloxía de desenvolvemento seguida no proxecto. Tamén se describe como se levou a cado a xestión da configuración e unha análise breve dos custos relacionados.

Capítulo 3. Análise

Neste capítulo descríbese o proceso de análise realizado, incluíndo a análise de requisitos e unha breve análise de tecnoloxías. Inclúense tamén as restriccións do proxecto.

Capítulo 4. Planificación

Exponse a planificación temporal, incluíndo a estimación temporal das actividades a realizar.

Capítulo 5. Desenvolvemento

Neste capítulo detállase como foi o proceso de desenvolvemento real do proxecto nas súas iteracións. Expóñense as distintas etapas, dando conta das tarefas realizadas en cada unha, e finalmente faise unha comparativa entre a planificación temporal inicial e como foi o desenvolvemento real.

Capítulo 6. Deseño e implementación

Neste capítulo descríbese a arquitectura do videoxogo e o deseño a alto nivel da aplicación. Expóñense detalles da implementación de varias das funcionalidades requeridas.

Capítulo 2

Xestión do proxecto

Neste capítulo abordaremos as actividades a planificar e levar a cabo ao longo da vida do proxecto. Faremos a elección da metodoloxía de desenvolvemento en función das particularidades do mesmo e dos obxectivos definidos anteriormente. Exporase a xestión da configuración realizada, comentando cales son os elementos de configuración e como se xestionaron. Finalmente, detállase a análise de custos asociados ao proxecto.

2.1. Metodoloxía de desenvolvemento

As metodoloxías de desenvolvemento definen o proceso de desenvolvmento do software. A elección da metodoloxía a seguir é unha decisión fundamental para o proxecto, dado que condicionará o seu desenvolvemento e en maior ou menor medida, o seu éxito ou fracaso. Polo tanto, seleccionar a metodoloxía axeitada para un proxecto proporcionará un marco de traballo que facilitará o seu éxito.

No ámbito das metodoloxías de desenvolvemento existen dúas correntes principais [2]. Por un lado están as **metodoloxías tradicionais**, que se centran no control dos procesos de desenvolvemento e un seguimento riguoso das tarefas asociadas. Este tipo de metodoloxías baséanse no seguimento estrito dun plan de proxecto, pouco flexible

e centrado en documentar todos os procesos. Por outra parte están as **metodoloxías áxiles**, centradas no factor humano, na colaboracion e na participación do cliente no proceso de desenvolvemento. Caracterízanse por incrementos contínuos do software, con iteracións curtas.

Tendo en conta as circunstancias nas que se desenvolve este proxecto, o alumno non conta cunha gran experiencia na xestión de proxectos, polo que unha metodoloxía tradicional parece pouco axeitada, dado que require ter experiencia na xestión. Ademais, o proxecto estará sometido a cambios ao longo do seu desenvolvemento, debido a que inicialmente non todos os requisitos están claramente definidos e hai unha alta probabilidade de que aparezan novos requisitos ao longo do proxecto, polo que a excesiva rigurosidade e a pouca flexibilidade das metodoloxías tradicionais non resultan axeitadas. Por último, é importante dispoñer dunha versión funcional do produto coa maior celeridade, posto que ao non ter claros os requisitos, ter unha versión funcional permite guiar o desenvolvemento en conxunción cos clientes. Cunha metodoloxía tradicional non se dispón dunha versión funcional do produto ata unha fase avanzada do desenvolvemento, polo que tampouco resultan axeitadas neste sentido.

Con este contexto, parece claro seguir unha metodoloxía áxil. Este tipo de metodoloxías tratane de aliviar o peso dos seus procesos defendendo a renuncia expresa a utilizar modelos perfectos, buscando únicamente que sexan suficientemente bos. Priorízase a funcionalidade sobre a documentación, ao individuo e as interaccións do equipo sobre os procesos e as ferramentas, e a colaboración co cliente e a resposta rápida ante cambios. Os principios das metodoloxías áxiles pódense consultar no *Manifesto Áxil* [3]. Dentro das metodoloxías áxiles optaremos por seguir **Scrum**, xa que permitiranos controlar e reducir a probabilidade de aparición de determinados riscos grazas aos mecanismos que define. A continuación exponse como se aplicou Scrum neste proxecto.

2.1.1. Aplicación de Scrum

Scrum ten unha coñecda guía [4] na cal se define coma unha metodoloxía liviana, fundamentada sobre o empirismo asumindo que un problema non se pode definir de todo. Céntrase en maximizar a habilidade do equipo de desenvolvemento para responder a requisitos novos ou cambiantes, un punto fundamental para este proxecto. Cabe destacar que non se seguiu Scrum ao pé da letra, senón que se adaptou ás necesidades específicas do proxecto. Para isto botamos man dos *ScrumButs* [5], que permiten definir regras e excepcións para que non se convirta nun problema seguir Scrum. Están indicados para situacións nas cales o equipo non pode aproveitar todas as vantaxes de Scrum. Un *ScrumBut* ten a seguinte sintaxe: **(ScrumBut)(Razón)(Solución)**.

O marco de traballo que define Scrum consiste no equipo de traballo cos seus roles asociados, eventos, artefactos e regras. Agora veremos os distintos elementos definidos e como se aplicaron e adaptaron ao proxecto:

Roles

Scrum define unha serie de roles, que en conxunto forman o equipo de traballo. Dito equipo debe ser multidisciplinar, autoxestionado e autoorganizado. O modelo de equipo está deseñado para optimizar a flexibilidade, creatividade e productividade. Os roles definidos son os seguintes:

- *Product owner*: representa aos *stakeholders*[1]. É a persoa única encargada de administrar o *product backlog*, que explicaremos máis adiante. Redacta as historias de usuario e engádeas ordenadas por prioridade ao *product backlog*.
  No caso deste proxecto, este rol foi asumido polos titores colaboradores de ESF, Miguel Rodríguez-Segade e Sergio Fernández, posto que son os clientes e

[1]Stakeholder: término en inglés utilizado por primeira vez por R.E. Freeman na súa obra *Strategic Management: A Stakeholder Approach*, Pitman, 1984, para referirse a quenes poden afectar ou son afectados polas actividade dunha empresa.

son os que definen os requisitos, que logo se traducen en historias de usuario.

- *Scrum master*: encárgase de que o equipo siga a teoría de Scrum, coas súas prácticas e regras. O seu traballo principal é o de eliminar os obstáculos que impiden que o equipo alcance os seus obxectivos. Axuda ao equipo a entender a necesidade de facer historias de usuario claras e concisas, e facilita os eventos de Scrum segundo sexa necesario.
  Este rol foi asumido polo alumno e polo titores en conxunto, participando conxuntamente na elaboración das historias de usuario e convocando e facilitando as reunións necesarias.

- Equipo de desenvolvemento: consiste no conxunto de profesionais encargados de entregar o produto. É un equipo autoorganizado e os seus membros deben ter habilidades transversais, con todos os coñecementos necesarios para crear os incrementos.
  O equipo de desenvolvemento neste proxecto está íntegramente formado polo alumno, que posúe os coñecementos necesarios e leva a cabo o traballo para crear cada incremento.

Eventos

Os eventos empréganse en Scrum para crear regularidade e minimizar a necesidade de reunións que non estean definidas. Os eventos contemplados son os seguintes:

- *Sprint*: é a unidade básica de desenvolvemento en Scrum. Un *Sprint* ten unha duración fixa de entre unha semana e un mes, sendo definida a duración en avance e non se pode modificar posteriormente. Durante o *Sprint* non se poden facer modificacións que poñan en perigo os obxectivos do mesmo, pero o alcance pode ser renegociado entre o *product owner* e o equipo de desenvolvemento. Ao final dun *Sprint* dispónse dun incremento do produto potencialmente entregable. Os *Sprints* actúan como contedores do resto de eventos que veremos a continuación.

Neste proxecto, os *Sprints* realizáronse en iteracións de entre dúas e tres semanas de duración, incluíndo unha serie de funcionalidades que permitían obter un novo incremento do produto incluíndo novas funcións e correccións de erros.

- *Sprint planning*: reunión de planificación dun *Sprint*. Faise ao inicio de cada *Sprint*, seleccionando o traballo a realizar e creando o *Sprint backlog*, que consiste no conxunto de ítems do *product backlog* que se inclúen no *Sprint*. O equipo de desenvolvemento ten que valorar o traballo a realizar e estimar o tempo que levará realizalo.
  Estas reunións levaronse a cabo de xeito habitual, realizándose moitas veces xusto ao finalizar a revisión do *Sprint* anterior (*Sprint review*).

- *Daily Scrum*: consiste nunha reunión diaria do equipo de desenvolvemento para sincronizar as súas actividades e planificar o traballo das próximas 24 horas.
  No caso deste proxecto non se realizou esta actividade, xa que o equipo de desenvolvemento consta dunha soa persoa polo que carece de sentido.

- *Sprint review*: este evento ten lugar ao final de cada *Sprint*. O equipo de desenvolvemento revisa o traballo que foi completado no incremento e o que non, e discute sobre o que foi ben no *Sprint*, e os problemas que se atoparon e cómo se resolveron. O traballo completado preséntase aos *stakeholders* e o *product owner* verifica os ítems que foron completados do *product backlog*. O resultado desta reunión é un *backlog* revisado que define os posibles ítems para o próximo *Sprint*.
  Esta reunión levouse a cabo de forma normal, sendo presentado o traballo realizado aos titores por parte do alumno ao rematar cada *Sprint*. Revisábase o traballo realizado, verificando e validando as funcionalidades implementadas, e discutíase e definíase o traballo a realizar no seguinte *Sprint*.

- *Sprint retrospective*: Nesta reunión o equipo de desenvolvemento reflexiona sobre o *Sprint* recentemente superado acordando accións de mellora continua

do proceso.

Neste proxecto esta reunión carece de sentido cunha soa persoa no equipo de desenvolvemento, polo que non se levou a cabo por consideralo unha perda de tempo.

Artefactos

- *Product backlog*: é a lista ordeada dos requerimentos do produto. O *product owner* encárgase da súa xestión, estando constituída por funcionalidades, *bugfixes*, requisitos non funcionais, etc. É dinámico e nunca está completo, modificándose contínuamente mentres exista o produto. A medida que avanza o desenvolvemento vaise refinando, engadindo maior detalle e mellores estimacións aos ítems.

  No noso proxecto tivose en conta en cada momento o conxunto de funcionalidades, *bugfixes* a realizar, que se iban priorizando en cada unha das reunións de seguimento realizadas.

- *Sprint backlog*: É o conxunto de ítems do *product backlog* que se seleccionan para un *Sprint*. O equipo de desenvolvemento encárgase de crear esta lista collendo elementos da cima do *backlog* ata que considere que ten suficiente traballo para o *Sprint*.

  Para cada *Sprint* realizado, fíxose unha lista co conxunto de funcionalidades e tarefas a realizar entre o alumno e os titores, en función da dispoñibilidade temporal e do traballo que se consideraba que se podía realizar en dito *Sprint*.

- Incremento: É a suma de todos os ítems do *product backlog* completados durante un *Sprint* e todos os *Sprints* anteriores.

2.2. Xestión da configuración

A xestión da configuración trata de identificar, organizar e controlar as modificacións do software construído por un equipo de desenvolvemento. O obxectivo é maximizar a productividade e minimizar os erros [6]. As actividades da xestión da configuración desenvólvense para identificar os cambios, controlalos, garantir que os cambios se implementen adecuadamente e notificalos a outras partes interesadas.

Liña base

Unha liña base é un concepto da xestión da configuración que axuda a controlar os cambios sen impedir cambios xustificables. No contexto da enxeñaría do software, unha liña base é un fito no desenvolvemento. Pódese marcar unha liña base para a entrega dun incremento que foi aceptado, por exemplo. Cando se quere modificar un elemento que se converteu en liña base, utilízase unha copia do proxecto no espazo de traballo privado da persoa que leva a cabo a modificación.
No caso deste proxecto, estableceuse coma liña base cada incremento do software creado en cada *Sprint*.

Elementos de configuración do software

Considérase coma elemento de configuración do software a información que se crea como parte do proceso de enxeñaría do software. Estos elementos débense de identificar e organizar para formar obxectos de configuración.
Para este proxecto, identificáronse os seguintes elementos de configuración:

1. **Código fonte**: Inclúese o código fonte da aplicación, os arquivos de configuración, directivas de compilación e o software empaquetado.

2. **Documentación**: Inclúe os ficheiros necesarios para poder xerar esta memoria, incluíndo as figuras, imaxes, diagramas, etc.

Depósito de elementos de configuración

É o repositorio onde se almacenan os elementos de configuración. Debe promover un conxunto de mecanismos e estruturas de datos que permitan manexar os cambios de forma eficaz.
Neste proxecto empregouse **Git** [7], un sistema distribuído de control de versións para o desenvolvemento de software. En concreto utilizouse ***GitLab***, que é un xestor de repositorios Git baseado na web con funcionalidades de seguimento de incidencias. Creouse un repositorio público, dispoñible durante todo o desenvolvemento, que se pode consultar no seguinte enlace:

`https://gitlab.com/daniel.barbeira/blockrain.js`

As modificacións correspondentes as cada *Sprint* desenvolvéronse nunha copia en local do repositorio, e a medida que se completaban funcionalidades sincronizábase a copia local co repositorio remoto. Cada vez que se remataba un *Sprint* tamén se sincronizaba o respositorio remoto. Deste xeito, pódese descragar a última versión da aplicación en calquera momento.

Na Figura 2.1 obsérvase unha gráfica que representa o número de *commits* realizadas contra o repositorio de *GitLab* ao longo dos meses de desenvolvemento. Pode observarse unha subida gradual ao principio, cando se comezou co proxecto e se desenvolveu un prototipo inicial. Esta tendencia alcanza un pico a finais de decembro/principios de xaneiro, momento no cal finaliza o desenvolvemento dese prototipo inicial.

Posteriormente, obsérvase que a actividade entre xaneiro e febreiro foi menos frecuente, debido ao tempo dedicado á elaboración do anteproxecto e un periodo de análise e búsqueda dalgunhas tecnoloxías e librerías necesarias para o desenvolvemento.

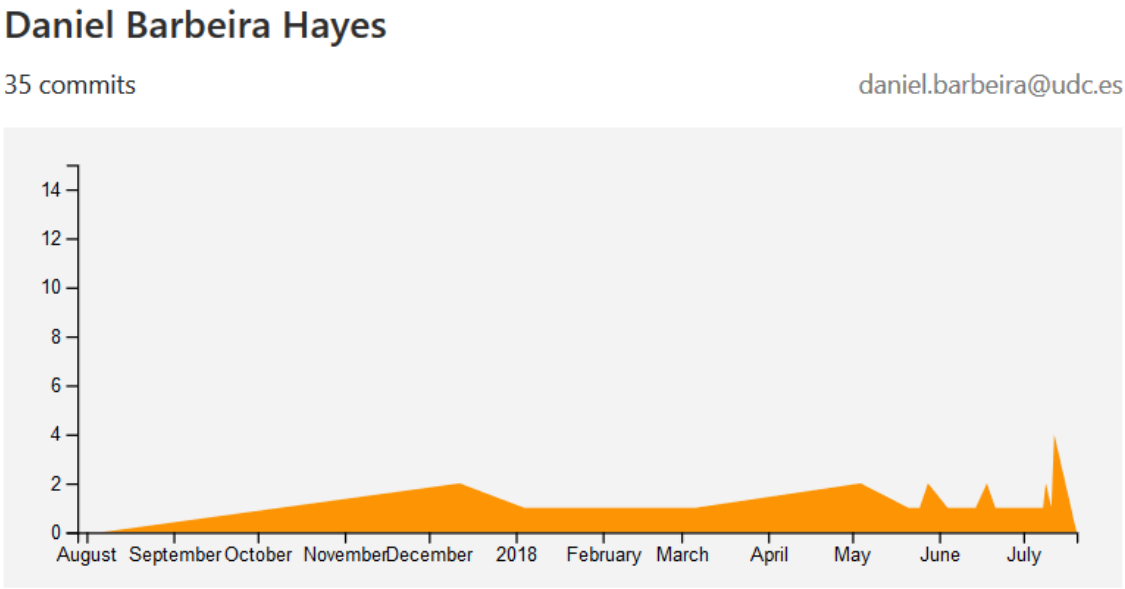

Figura 2.1: Contribucións (*commits*) ao repositorio *GitLab*.

A partir do mes de marzo, obsérvase unha maior actividade que se estende ata finais de xullo/ agosto, coincidindo co periodo temporal onde se realizaron as principais iteracións no desenvolvemento do videoxogo.

2.3. Análise de custos

A xestión dos custos dun proxecto inclúe procesos de planificación, estimación, preparación dun presuposto e control de custos de forma que o proxecto poida completarse dentro do presuposto aprobado. Ocúpase principalmente do custo dos recursos necesarios para completar as actividades planificadas, pero tamén debe ter en conta os custos de uso, mantemento e soporte do produto [8, *Capítulo 7*].

Hai que ter en conta que este proxecto trátase dun Traballo Fin de Mestrado, e polo tanto os custos que se estimarán son teóricos. Para realizar a estimación de custos, diferenciamos entre os custos relativos ao persoal e os custos materiais e indirectos.

2.3.1. Custos do persoal

Consideramos os custos relativos aos membros do equipo de desenvolvemento, que neste caso trátase de soamente un, o alumno. Os titores asumiron un rol de clientes. O alumno asumiu un rol de analista/programador. Tomamos como referencia para calcular os custos a guía de salarios do sector das TI en Galicia elaborada por *Vitae Consultores* [9].

Nesta guía pódese observar que o salario medio dun analista/programador junior é de 16000€ anuais. Considérase que de media hai un total de 148 días non laborables no ano entre festivos, fins de semana e vacacións. Con isto, tendo en conta os 217 días laborais do ano e unha xornada laboral diaria de 8 horas, o cálculo do salario medio por hora danos un total de:

$$16000/217/8 = 9{,}22€ \textit{ por hora}$$

O número de horas totais dedicadas ao proxecto está estimado segundo o número de créditos da materia do Traballo Fin de Mestrado (18 créditos) e o número recomendado de horas por crédito (25 horas). Con isto temos unha estimación de 450 horas, polo que os custos de persoal ascenden a un total de **4149€**.

2.3.2. Custos materiais e indirectos

O material necesario para levar a cabo o proxecto consiste basicamente nun computador, con todo o software necesario correctamente instalado e configurado. O software non supón ningún custo adicional, dado que se empregou software libre e gratuíto. Como o alumno xa dispoñía dun computador portátil persoal, tampouco suporá ningún custo para o proxecto. O custo do material de oficina empregado estímase en 5€ de gasto asociado.

Como custo indirecto terase en conta o consumo eléctrico do equipo de traballo. Coma o prezo do kWh varía de forma contínua, empregamos o prezo medio do ano

2018 que está fixado en 0,12688 €/kWh no mercado regulado [10]. A corrente eléctrica que consume un computador portátil é variable en función das súas características e estímase en 60W de media [11]. Para calcular o custo do consumo eléctrico derivado da utilización do computador, primeiro calcúlanse os kWh en total que consume e logo multiplícase esa cifra polo custo do kWh:

$$60\,\mathrm{W} \times 450\,\mathrm{h} = 27000\,\mathrm{Wh}/1000 = 27\,\mathrm{kWh}$$

$$27\,\mathrm{kWh} \times 0{,}12688\,€/\mathrm{kWh} = 3{,}43\,€$$

No seguinte cadro amósase a desagregación dos custos do proxecto xunto co custo total:

Concepto	**Importe**
Salario do persoal	4149 €
Material de oficina	5 €
Consumo eléctrico computador portátil	3.43 €
Total	4157.43 €

Cadro 2.1: Desagregación de custos

Capítulo 3

Análise

Neste capítulo exporemos as tarefas de análise realizadas no proxecto, coa finalidade de obter unha especificación o máis completa posible. Detallaremos a problemática tratada no videoxogo, a análise dos requisitos xunto coa definición das restricións e, finalmente, a análise das tecnoloxías e ferramentas empregadas no desenvolvemento.

3.1. Análise da problemática

O persoal de ESF destaca a seguinte carencia a tratar:

"En xeral, falta coñecer en profundidade por parte da base social da asociación a parte filosófica do concepto de Tecnoloxías para o Desenvolvemento Humano, que ademais de ser a especialización da asociación, une conceptualmente os distintos temas que se traballan (dereito á auga, dereito á alimentación e soberanía alimentaria, soberanía dixital, soberanía enerxética, feminismos...) para ter un discurso coherente."

Cando se trata de abordar esta carencia, estamos ante un contexto de falta de

tempo das persoas da base social para formarse nun tema que non é sinxelo de transimitir, cunha curva de aprendizaxe ardua. Aínda así existe unha forte motivación na base social para dedicar o seu tempo a lograr un mundo mellor.

Dado que o público obxectivo serían principalmente persoas socias voluntarias, traballadores e colaboradores cercanos á asociación (e, en segunda instancia persoas socias non voluntarias ou outras interesadas na tecnoloxía e nos dereitos humanos), serán persoas interesadas pola tecnoloxía e por aportar o seu "gran de area" para conseguir un mundo máis xusto e solidario. Con isto, propúxose o seguinte obxectivo xeral:

"Aumentar o coñecemento sobre o concepto de Tecnoloxías para o Desenvolvemento Humano na base social e persoas colaboradoras de ESF Galicia. "

Para lograr este obxectivo, propúxose o desenvolvemento dun videoxogo baseado no coñecido *tetris* a partir dun dos numerosos proxectos libres que implementan un *tetris*, para adapatalo ás necesidades mantendo a licenza libre. Dado que o videoxogo *tetris* é parte da cultura popular, deséxase explorar o potencial para atraer audiencias interesadas en xogos dándolle un contido sensibilizador no eido dos Dereitos Humanos. Desta forma téntase sensibilizar ás persoas que poden xogar cun produto elaborado. Ademais, a idea é que se poida xogar tanto desde un PC coma nun dispositivo móbil.

Unha das parte que máis hai que traballar é a visibilización e integración dos contidos de Dereitos Humanos, xa que en realidade, o *tetris* é un videoxogo que obvia a construción dunha historia que o sustente. Simplemente é "facer un puzzle", sen máis.

A idea fundamental é que as pezas a encaixar, que teñen diversas formas e cores teñan asociados segundo a súa forma e cor as seguintes variables:

- **A cor da pezas**: será distintivo do servio básico vinculado a un Dereito Humano. Trabállase co dereito á auga en azul, dereito á enerxía en amarelo, dereito á información e comunicacións en laranxa e o dereito á alimentación

en verde. Ademais incluirase un dereito transversal de igualdade de xénero (en violeta).

- **A forma das pezas**: axustarase á facilidade (pezas máis sinxelas) ou á ameaza (pezas difíciles de encaixar) para asegurar un servizo básico. Por exemplo, a lexislación axeitada ou a administración pública eficiente serían figuras sinxelas de encaixar coma as cadradas; formas en "L" poden corresponder a infraestructuras deficientes e, casos de corrupción pode vir na forma de "Z".

Outra idea importante é a de transmitir a existencia de condicionantes máis alá dos tecnolóxicos á hora de asegurar un Dereito Humano. Para iso é importante visualizar as diferenzas existentes entre países para asegurar un Dereito Humano e como en cada país se aborda dunha forma distinta. Desta forma, en cada país será necesario incluír unha pantalla con cada Dereito Humano traballado e unha última onde saian pezas con todos os dereitos (cores) mezclados. Unha vez superado pasaríase ao seguinte país (incrementando a dificultade progresivamente).

Utilizar un videoxogo, algo relacionado directamente coa tecnoloxía moderna, e conseguir visiblizar e destacar que se trata dun proxecto de software libre e educativo sobre Dereitos Humanos, espérase que o producto sirva de "imaxe simbólica de marca" da organización ESF, cuxa misión é precisamente poñer a tecnoloxía ao servicio do desenvolvemento e os Dereitos Humanos.

3.2. Análise de requisitos

Nas metodoloxías áxiles empréganse as historias de usuario coma formato para especificar os requisitos. As historias de usuario consisten nunha descrición curta e sinxela pero suficiente para definir un requisito. Para expresalas dunha maneira formal, pódense redactar seguindo o seguinte formato:

Como (rol) quero (algo) para poder (beneficio).

Este formato permite responder ás seguintes cuestións: *Quen se beneficia?*, *Que se quere?* e *Cal é o beneficio?* A lista ordeada por prioridade das historias de usuario constitúe o *product backlog*. Utilizaremos un formato de ficha para redactar as historias, incluíndo un identificador para cada historia, a descrición e os criterios de aceptación.

3.2.1. Historias de usuario

Ao inicio do proxecto, nas primeiras reunións, discutíronse e definíronse as características e funcionalidades iniciais do videoxogo a desenvolver. A continuación está o listado dos requisitos iniciais que se definiron a partir desas reunións e das ideas que se discutiron nelas. Os requisitos que apareceron posteriormente durante os *Sprints* do desenvolvemento definirémolos no capítulo de desenvolvemento correspondente.

Título: Interacción co videoxogo
Identificador: HU.1
Descrición: Como usuario quero poder interaccionar co videoxogo mediante os controis habituais e coas mecánicas dun tetris clásico e que o xogo se comporte de forma consistente, para poder interactuar con él e avanzar nos niveis definidos, alcanzando as puntuacións definidas.
Criterio de aceptación: Un usuario que acceda ao videoxogo, pode interactuar con él cos controis do teclado, podendo iniciar unha partida e que o xogo controle o fluxo e a interacción co usuario guiándoo polos distintos niveis.

Cadro 3.1: Historia de Usuario 1

Título: Cores das pezas
Identificador: HU.2
Descrición: Como administrador quero definir as cores das pezas para diferenciar os distintos servizos básicos vencellados a un dereito humano. En función do dereito humano tratado en cada nivel, as cores son azul para a auga, verde para a alimentación, amarelo para a enerxía e laranxa para as TIC.
Criterio de aceptación: O administrador poderá parametrizar nun ficheiro de configuración as cores das pezas nun nivel, para así crear distintos niveis relacionados cos distintos dereitos humanos. As pezas deben saír nas mesmas cores que as definidas no ficheiro.

Cadro 3.2: Historia de Usuario 2

Título: Textos e dificultade por pezas
Identificador: HU.3
Descrición: Como administrador quero definir os distintos textos informativos relacionados con cada tipo de peza para visibilzar contidos sensibilizadores no ámbito dos dereitos humanos. O tipo de peza determínase en función da dificultade para encaixar dita peza.
Criterio de aceptación: No videoxogo, ao caer a peza debe aparecer na parte superior da pantalla e na mesma cor o texto informativo relacionado co dereito humano que se está tratando. O criterio da dificultade para encaixar unha peza é definida polo administrador. Desta forma, preténdese que o usuario sexa consciente de porqué é máis ou menos complicado de encaixar a peza en cuestión.

Cadro 3.3: Historia de Usuario 3

Título: Pezas personalizadas
Identificador: HU.4
Descrición: Como administrador quero que o videoxogo sexa capaz de xerar pezas con formas personalizadas, distintas do xogo de pezas clásico dun tetris, para poder definir tanto pezas máis sinxelas como máis complicadas e poder darlle un maior dinamismo e unhas dificultades adicionais ao videoxogo.
Criterio de aceptación: O videoxogo debe xerar as pezas definidas cunha distribución que se define na configuración, e ditas pezas deben de encaixar e ter un comportamento equivalente ao resto de pezas.

Cadro 3.4: Historia de Usuario 4

Título: Xeración de obstáculos adicionais
Identificador: HU.5
Descrición: Como administrador quero poder configurar o videoxogo para que antes de comezar unha partida poidan aparecer obstáculos prefixados, para engadir unha maior dificultade nos distintos niveis.
Criterio de aceptación: No xogo deben aparecer antes de iniciar cada nivel unha serie de casillas xa ocupadas. A distribución das casillas ocupadas será aleatoria e o número de casillas que saian enchidas debe poder definirse nun ficheiro de configuración.

Cadro 3.5: Historia de Usuario 5

Título: Definición dos niveis do xogo
Identificador: HU.6
Descrición: Como administrador quero que o videoxogo teña distintos niveis de dificultade para poder amosar as diferenzas no estado dos dereitos humanos en distintos países do mundo.
Criterio de aceptación: No videoxogo os niveis definiranse en función do país que se está a tratar. Por cada país haberá catro subniveis, un por cada dereito humano tratado (auga, alimentación, enerxía e TICs).

Cadro 3.6: Historia de Usuario 6

Título: Nivel mixto
Identificador: HU.7
Descrición: Como administrador quero definir un nivel no cal poidan aparecer textos e pezas de todas as temáticas de dereitos humanos tratados, para poder establecelo como subnivel final de cada país e integrar as distintas temáticas tratadas en cada un dos subniveis do país.
Criterio de aceptación: Logo de completar os subniveis correspondentes a cada dereito humano nun país, cargarase un subnivel final no cal aparecen pezas misturadas de cada un dos distintos dereitos humanos, é dicir, poden saír pezas azuis, laranxas, amarelas ou verdes.

Cadro 3.7: Historia de Usuario 7

Título: Notificacións ao usuario
Identificador: HU.8
Descrición: Como administrador quero xerar notificacións para o usuario cando ocorren determinadas accións no videoxogo para controlar o fluxo e incrementar a interacción do usuario.
Criterio de aceptación: O videoxogo debe xerar notificacións ante determinados eventos que ocorran como o inicio ou o fin dun nivel. Débese poder establecer o texto da notificación, botóns de interacción, duración, etc.

Cadro 3.8: Historia de Usuario 8

Título: Internacionalización
Identificador: HU.9
Descrición: Como usuario quero que o videoxogo se poida xogar no idioma da miña preferencia para que a interacción sexa o máis natural posible e para poder entender correctamente os textos do videoxogo.
Criterio de aceptación: O videoxogo disporá dun selector que permita elixir o idioma. Os idiomas nos cales os textos están traducidos son o galego e o castelán, sendo o galego o idioma por defecto. Ao recargar a páxina débese manter o idioma seleccionado.

Cadro 3.9: Historia de Usuario 9

Título: Caída rápida (*Quickdrop*)
Identificador: HU.10
Descrición: Como usuario quero poder facer que a caída dunha peza sexa repentina para poder xogar de forma máis rápida e áxil.
Criterio de aceptación: O usuario que está xogando debe poder executar unha caída repentina da peza ao pulsar a barra espaciadora.

Cadro 3.10: Historia de Usuario 10

Título: Integración dun mapa do mundo
Identificador: HU.11
Descrición: Como usuario quero poder navegar entre os distintos niveis (países) incluídos no videoxogo para poder avanzar e chegar ata o último nivel para completar todos os niveis.
Criterio de aceptación: Para navegar entre os distintos niveis, o videoxogo debe cargar un mapa do mundo cos países desbloqueados resaltados nunha cor (verde) e os países por desbloquear noutra (vermello). Ao facer clic nun país cargarase o nivel de dito país cos seus subniveis correspondentes.

Cadro 3.11: Historia de Usuario 11

Título: Parametrización dos niveis e grafo do xogo
Identificador: HU.12
Descrición: Como administrador quero ter unha estrutura semellante á dun grafo dirixido para poder implementar o fluxo de control do videoxogo. Cada nodo representa un nivel cos seus subniveis correspondentes e en cada nodo débese poder parametrizar características do nivel e os seus subniveis.
Criterio de aceptación: No videoxogo debe incluirse unha funcionalidade para parametrizar os niveis/subniveis nun ficheiro de configuración. Cada ficheiro de configuración debe incluír unha referencia ao seguinte ficheiro a cargar, creando así unha relación entre os distintos ficheiros o cal permitirá simular un grafo dirixido.

Cadro 3.12: Historia de Usuario 12

3.2.2. Restricións

Na fase inicial do proxecto definíronse unha serie de restricións ás que está suxeito, por requisitos do cliente ou ben pola autoimposición de seguir unha filosofía de software libre. As restricións definidas son as seguintes:

1. **Multiplataforma**: o videoxogo a implementar debe ser multiplataforma. Débese de poder acceder e xogar tanto dende un PC coma dende distintos dispositivos móbiles, independentemente do sistema operativo se é posible.

2. **Software libre**: todas as tecnoloxías empregadas no desenvolvemento do proxecto serán gratuítas e de código aberto, dacordo coa definición de software libre da *Free Software Foundation* [1]. O sistema operativo non é limitante, podendo levarse a cabo o desenvolvemento sobre calquera plataforma.

3. **GitLab**: coma servizo de control de versións empregarase GitLab, sendo esta unha ferramenta publicada baixo unha licenza de código aberto (licenza MIT). Descártase a utilización de GitHub (o servizo de xestión de repositorios *Git* máis popular) polo feito de tratarse dunha empresa privada cuxo código non é libre contendo partes que son propietarias.

3.3. Análise de tecnoloxías

Nesta sección farase unha análise das tecnoloxías empregadas no desenvolvemento do videoxogo. Analizaremos as distintas tecnoloxías e proxectos libres que implementan un *tetris* para tomar como base do videoxogo a implementar, se é posible. Ao mesmo tempo faremos un estudo das distintas librerías necesarias para implementar as funcionalidades requeridas, e abordaremos as ferramentas necesarias para o desenvolvemento do proxecto.

3.3.1. Videoxogo *tetris* baseado en tecnoloxías web

Para implementar un videoxogo de tipo *tetris* baseado en tecnoloxías web, existen varios proxectos libres que se poden tomar como punto de partida. Posto que debe ser multiplataforma, será fundamental que a interface sexa adaptable a distintos tamaños de pantalla (*responsive design*), ademais de poder engadir os controis correspondentes para dispositivos móbiles. Por outra parte buscamos unha tecnoloxía que nos permita crear un videoxogo básico de forma rápida e que logo sexa sinxelo de estender e de engadir as librerías necesarias para implementar as funcionalidades requeridas.

Para a elección do proxecto que se utilizará como base tivéronse en conta os seguintes criterios:

- Licenza: O proxecto usado debe ter unha licenza de código aberta e de uso gratuíto.
- Extensibilidade: A facilidade de adaptar o código a cambios ou a novas funcionalidades será un factor importante a ter en conta. É de interese que dispoña dunha API que facilite engadir funcións novas de forma rápida e efectiva.
- Documentación: Valorarase de forma positiva que o proxecto escollido teña unha documentación clara e concisa, expoñendo as súas funcionalidades e con exemplos de utilización.
- Experiencia previa de uso: A experiencia previa de uso permitirá reducir os tempos de desenvolvemento do software e os prazos de entrega. Ademais, facilitará a construción dun software de maior calidade e fiabilidade.

Aplicando os criterios descritos, e tendo en conta as propiedades dos distintos proxectos analizados e os requerimentos do videoxogo, optouse por utilizar o seguinte proxecto libre:

`https://github.com/Aerolab/blockrain.js/`

O proxecto *blockrain.js* cumpre con cada un dos criterios definidos ademais de ser a alternativa máis completa e actualizada, e a mellor documentada. O proxecto consiste nunha implementación dun *tetris* escrito en HTML5 + CSS3 e JavaScript. Está configurado coma un plugin de jQuery o cal permite definir métodos sobre un mesmo selector de jQuery que executan unha serie de accións sobre dito selector. Esta configuración é moi útil para cumprir co criterio da extensibilidade, xa que permite engadir novas funcións de forma sinxela.

Para aproveitar as posibilidades ofrecidas polo código do proxecto *blockrain.js*, creouse un *fork* (bifurcación) en *GitHub* como unha copia exacta do código orixinal. A URL de dito *fork* é a seguinte:

`https://github.com/danibarbeira/blockrain.js`

Posteriormente importouse dito *fork* ao repositorio creado en ***GitLab*** para o presente proxecto.

3.3.2. Librerías adicionais

Para desenvolver o videoxogo *tetris* empregando tecnoloxías web, ademais dos obxectos e funcionalidades estándar de JavaScript, é necesario facer uso dunha serie de librerías externas que aportan a funcionalidade extra requerida. Necesitamos librerías para poder resolver os seguintes puntos:

- **Automatización de tarefas**: Moitas das tarefas de desenvolvemento diarias que son repetitivas poden ser simplificadas si se automatizan. A automatización permite aforrar tempo e mellorar a productividade. *Gulp.js* [12] é un conxunto de ferramentas utilizado para executar tarefas de JavaScript de forma automatizada coma a minificación e concatencación do código, optimización, análisis de código ou copiar ficheiros modificados a un directorio de saída.

  Coma o proxecto *blockrain.js* inclúe unha configuración básica de *Gulp.js* e tendo en conta as súas vantaxes decidiuse manter e adaptar a configuración

para as necesidades do noso proxecto.

- **Compilación**: Dende o punto de vista do desenvolvemento, existe a necesidade de crear un código cohesionado e cun baixo acoplamento entre os compoñentes. Desta forma créase un software fácilmente extensible, modular e flexible. Para logralo é fundamental poder crear módulos independentes e illados que encapsulen as funcionalidades e que logo se poida exportar/importar ditos módulos uns noutros.

  En JavaScript existen varias ferramentas para conseguir crear módulos no lado cliente e utilizar a palabra clave *require* para importalos, da mesma forma que se pode facer con Node.js no lado servidor. As máis populares son *RequireJS*, *Browserify* ou *Webpack*. Optouse por utilizar *Browserify* [13] por ser máis sinxelo de configurar que *Webpack* e máis moderno e sinxelo de utilizar que *RequireJS*.

- **Internacionalización i18n**: Para poder ter os textos da aplicación en múltiples idiomas, necesitamos unha librería de JavaScript que nos perimta traballar coa internacionalización i18n. Existen algunhas alternativas coma *i18next* ou *jQuery.i18n*.

  Optouse pola utilización da librería *jQuery.i18n* [14] pola súa configuración sinxela e por ter directamente integrado a carga dos ficheiros cos textos traducidos (en formato JSON).

- **Notificacións**: Precísase unha librería que permita crear notificacións configurables e máis atractivas esteticamente e funcionais que un simple *alert* de JavaScript.

  A mellor alternativa atopada e finalmente utilizada é a librería *SweetAlert2* [15]. Permite crear *pop-ups* adaptables e configurables sen ningunha dependencia adicional.

- **Configuración**: Os ficheiros de configuración estarán definidos en formato

JSON (*JavaScript Object Notation*). É necesario poder cargar e extraer a información contida en ditos ficheiros para a súa utilización no videoxogo.

Para realizar a carga dos ficheiros optouse por empregar un obxecto *XMLHttpRequest* [16], que permite facer unha petición HTTP de tipo *get* que, pasándolle a ruta onde se atopa o ficheiro devolve unha referencia ao mesmo e podemos logo extraer a información dos distintos campos definidos no ficheiro.

- **Mapas**: Necesitamos unha librería que permita xerar un mapa do mundo no cal se poidan resaltar uns países determinados e ao picar nun país ou outro se poida realizar algunha acción controlada. En JavaScript existen algunhas alternativas coma *JSMaps*, *jVectorMap* ou *JQVMaps*.

  Finalmente optouse por utilizar *JQVMaps* [17], un proxecto que consiste nunha versión modificada e actualizada de *jVectorMap* (que leva algúns anos abandonada). Esta librería permite crear mapas altamente personalizables grazas á gran cantidade de opcións de configuración que inclúe.

3.3.3. Ferramentas

Nesta sección detallamos as ferramentas que se empregaron para o desenvolvemento do proxecto:

- **NPM**: Para poder instalar e xestionar as ferramentas baseadas en Node.js que utilizamos na nosa aplicación (*Gulp, Browserify*, etc) utilizamos NPM, o xestor de paquetes de *Node.js*. Para poder utilizalo é necesario ter unha instalación básica de *Node.js* que inclúe por defecto o xestor NPM.

  Con NPM, podemos especificar as dependencias do noso proxecto nun ficheiro *package.json* e, ao comezar a traballar no proxecto, con executar o comando `npm install` xa se instalan automáticamente todas as dependencias definidas. Tamén permite especificar as versións concretas necesarias para o proxecto en cuestión.

- **Visual Studio Code (VSCode)**: É un editor de código fonte desenvolvido por Microsoft, de código aberto e gratuito. Inclúe soporte para *debugging* ademais de soportar por defecto control de versións con *Git*, resaltado de sintaxe, etc.

  Dispón dunha gran cantidade de plugins para traballar coas tecnoloxías empregadas no proxecto. Para o desenvolvemento en JavaScript utilizaronse algunhas extensións coma *Babel JavaScript* para a comprobación de sintaxe e *Beautify* para formatear o código.

- **Servidor http**: Para poder cargar o videoxogo necesitamos acceder a él a través dun navegador web. Para iso, cargaremos a web do videoxogo nun servidor http, o cal permitirá tamén o acceso dende un dispositivo móbil. Optamos por utilizar *http-server*, un servidor http sinxelo de línea de comando e que non necesita ningunha configuración previa.

- **ShareLatex**: É un editor de LaTeX online que empregamos para a elaboración da documentación do proxecto. Das súas características destacamos a súa interface cómoda e sinxela de utilizar, o compilado online a formato PDF, o autocompletado e a posibilidade de colaborar con varios usuarios ao mesmo tempo.

- **Software de control de versións**: A solución escollida para o control de versións é *Git*. Aproveitaremos a integración directa que ten con VSCode, o cal permítenos aproveitar as posibilidades que ofrece *Git* de forma rápida e sinxela, coma a clonación de repositorios, a importación de proxectos dende un respositorio, realizar *commits*, etc. Tamén se utilizou a interface web de *GitLab*.

Capítulo 4

Planificación

Neste capítulo desenvolveremos a planificación temporal do proxecto. A metodoloxía *scrum* caracterízase polo solapamento das distintas fases do desenvolvemento, polo que en cada iteración ou *Sprint*, realízanse tarefas de análise, deseño, implementación e probas.

A estimación da duración total do proxecto é de 450 horas, que se corresponde con número de créditos contemplados na materia do Traballo Fin de Mestrado. Cun ritmo de traballo estimado de 25 horas semanais, prevese que se necesitarán un total de 18 semanas para completar o proxecto.

4.1. Definición de actividades

O primeiro paso para realizar correctamente unha planificación temporal é a identificación das actividades a levar a cabo. Descompoñemos o proxecto nun número manexable de actividades, que deben incluír tarefas de análise, desenvolvemento, xestión, etc.

Para iso, desenvolveremos un **EDT** (*Estrutura de Descomposición do Traballo*). Esta ferramenta consiste nunha descomposición xerárquica das actividades a realizar

no proxecto, onde cada nivel descendente representa unha definición máis detallada do traballo do proxecto. Como se pode ver na Figura 4.1, o proxecto estará dividido nunha serie de fases. Haberá unha fase de inicio, na cal estableceranse os obxectivos do proxecto, e farase un estudo inicial das posibilidades que se poidan abarcar. A continuación terá lugar unha fase de análise e planificación inicial, denominada *Sprint 0*, na cal crearase o *product backlog*, ademais de facer un estudo das tecnoloxías necesarias para levar a cabo o desenvolvemento e establecer a arquitectura do sistema. Logo comézase co desenvolvemento do videoxogo, durante o cal se levarán a cabo unha sucesión de *Sprints* correspondentes ás iteracións do producto. Finalmente terá lugar unha fase de finalización do proxecto, na que se realizará o despregamento do videoxogo e a documentación.

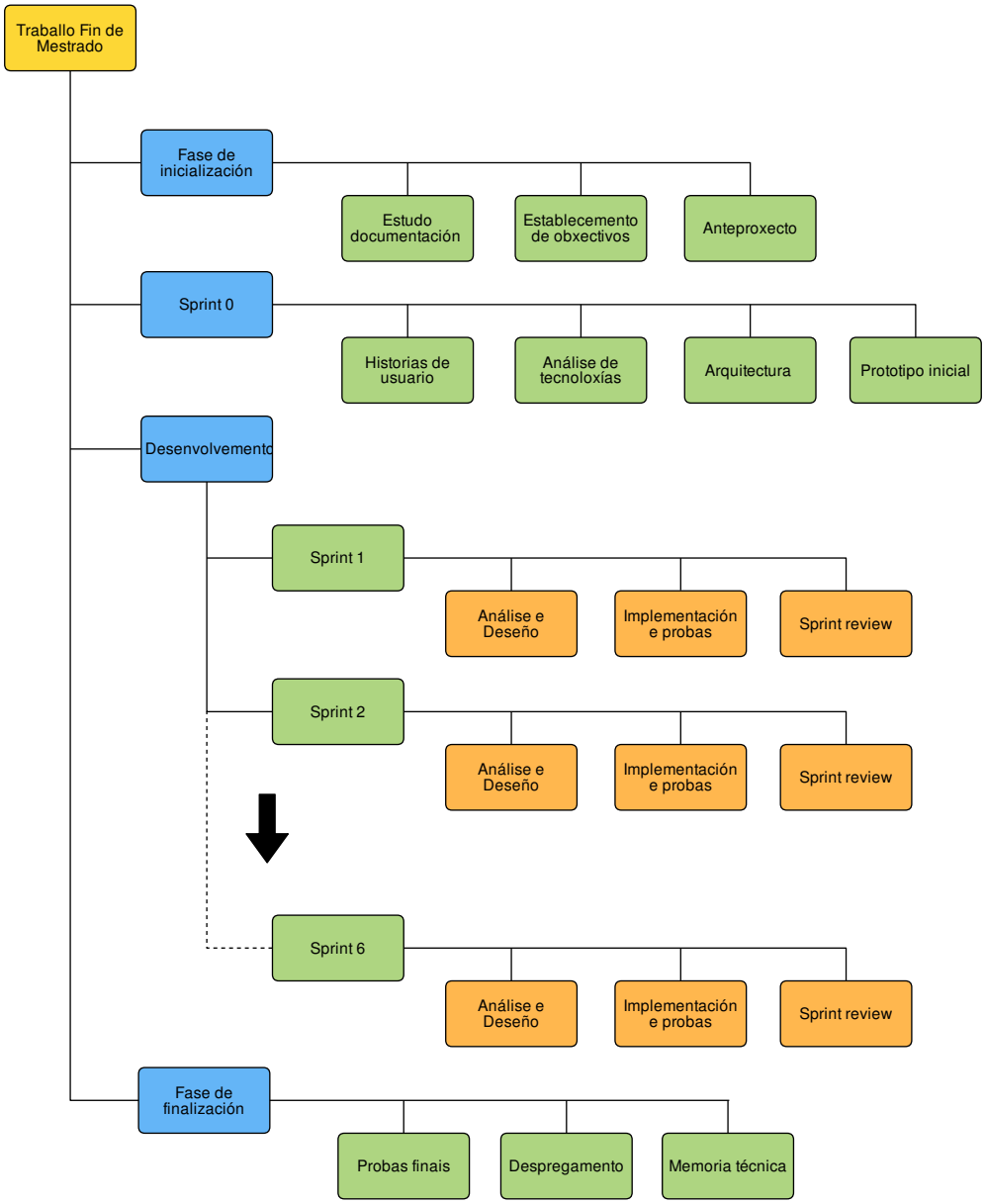

Figura 4.1: EDT do proxecto.

4.2. Estimación temporal das actividades

Neste apartado detallaremos cada unha das fases nas que se divide o proxecto, indicando a súa duración estimada e as actividades que se levarán a cabo.

4.2.1. Fase de inicialización

A fase de inicialización terá unha duración dunha semana. Durante este tempo levarase a cabo a definición do alcance do proxecto, o establecemento dos obxectivos e unha análise inicial do problema a abordar. Escollerase a metodoloxía de traballo a seguir e estableceranse as vías de comunicación cos titores do proxecto. Elaborarase o anteproxecto correspondente como resultado da análise inicial en conxunción cos titores.

4.2.2. Sprint 0

O *Sprint 0* está contemplado coma un *Sprint* especial, posto que non contempla actividades de desenvolvemento nin de revisión, senón que está centrado en tarefas de análise e organización. Esta *Sprint* especial terá unha duración de dúas semanas.

En primeiro lugar, realízase a análise inicial dos requisitos, redactando a historias de usuario. Priorízanse as historias e ordenanse en función da súa prioridade, formando o *product backlog* inicial. Unha vez redactadas as historias, será necesario levar a cabo unha análise das tecnoloxías necesarias para poder levar a cabo a implementación do videoxogo. Estudaranse distintas alternativas escollendo as que mellor se adapten ás funcionalidades requeridas e que cumpran coas restricións do proxecto. Nesta fase será na que se realice a análise de proxectos libres que implementan un *tetris* e, se é posible, tomar un como punto de partida para desenvolver o noso videoxogo. Finalmente definirase a arquitectura inicial do sistema, establecendo os compoñentes software necesarios, e crearase un prototipo sobre o que traballar no

primeiro *sprint*.

4.2.3. Fase de desenvolvemento

A fase de desenvolvemento estará dividida en 6 *Sprints*, cunha duración de dúas semanas cada un, seguindo as recomendacións de *Scrum*. Desta forma esta fase durará un total de 12 semanas. En cada *Sprint* seleccionaranse o conxunto de historias de usuario a incluír, conformando o *sprint backlog*. Farase unha análise e deseño breves, implementaranse e probaranse as funcionalidades requeridas e ao cabo de cada *Sprint* levarase a cabo a reunión de revisión e validación cos titores (o *Sprint review*).
A continuación exponse cada *Sprint* indicando as historias de usuario que se abordarán en cada un:

- ***Sprint* 1**: neste *Sprint* farase o deseño inicial do videoxogo, levando a cabo a implementación dun prototipo básico. Tamén se probarán algunhas das librerías para verificar que aportan a funcionalidade requerida. Implementaranse as funcionalidades de interacción co videoxogo e a asociación das cores das pezas cos dereitos humanos tratados.
  - Historias de usuario a abordar: HU.1, HU.2.
- ***Sprint* 2**: aboradarase a internacionalización do videoxogo, engadindo os textos traducidos ao galego e ao castelán. O idioma por defecto será o galego. Crearanse os arquivos de configuración dos niveis e implementarase a funcionalidade para cargar ditos ficheiros.
  - Historias de usuario a abordar: HU.9, HU.12.
- ***Sprint* 3**: neste *Sprint* levarase a cabo a funcionalidade das notificacións, engadindo a librería correspondente e implementando as accións de control das ventás modais. Engadiranse os textos asociados a cada peza e definirase a tipificación das pezas en función da dificultade para encaixar cada peza.

- Historias de usuario a abordar: HU.3, HU.8.

- ***Sprint* 4**: crearanse os niveis do xogo baseados nos países cos que se traballará, que serán España e Honduras. Por cada nivel (país) será necesario crear catro subniveis, un por cada dereito humano tratado no país. Os dereitos humanos abordados son a auga, a alimentación, a enerxía e o acceso ás TIC. Por outra parte, implementarase a función de caída rápida das pezas coa barra espaciadora.

  - Historias de usuario a abordar: HU.6, HU.10.

- ***Sprint* 5**: neste *Sprint* levarase a cabo a xeración de obstáculos, incluíndo a parametrización nos niveis e a xeración dos bloques fixados. Os bloques aparecerán antes de comezar o nivel e será posible encaixar o resto de pezas arredor deles para facer liñas. Incluirase tamén a páxina principal do videoxogo co mapa do mundo para navegar entre os distintos países.

  - Historias de usuario a abordar: HU.5, HU.11.

- ***Sprint* 6**: en último lugar desenvolveranse as funcionalidades restantes. Incluirase a posibilidade de xerar pezas con formas distintas ao xogo de pezas clásico dun *tetris*, sendo xeradas estas peza da mesma forma que as habituais. Engadirase tamén unha parametrización que permita establecer o porcentaxe de pezas dun tipo ou doutro que se xeren. Por outra parte, crearase o nivel mixto que inclúe pezas de distintas cores integrando as temáticas tratadas nos distintos subniveis.

  - Historias de usuario a abordar: HU.4, HU.7.

4.2.4. Fase de finalización

A fase de finalización terá unha duración de tres semanas. A maioría do tempo dedicarase á elaboración da presente memoria, que relata como foi o desenvolvemento completo do proxecto. Realizaranse as probas finais cos titores para validar as

funcionalidades implementadas e levarase a cabo o despregamento do videoxogo nos servidores de ESF.

Capítulo 5

Desenvolvemento

Neste capítulo detallarase como foi o desenvolvemento do proxecto. Exporanse os *Sprints* tal e como se desenvolveron, de acordo coa planificación temporal. Estableceuse unha duración de 15 días para cada *Sprint*, e tras o primeiro *Sprint* decidiuse manter esta duración ao longo de todo o proxecto. Inclúese tamén a especificación das historias de usuario que foron aparecendo durante o desenvolvemento.

5.1. Sprint 1

Neste *Sprint* realizouse de forma inicial a implementación dun prototipo básico do videoxogo. Este prototipo serviu como base do videoxogo a desenvolver, para probar algunhas das tecnoloxías e para dispoñer dunha primeira versión funcional. Esta versión inicial inclúe dous niveis, un de España e outro de Honduras. Definiuse a disposición dos elementos visuais básicos dunha partida, coma a puntuación, o nome do nivel ou o lugar onde se amosará o texto asociado a cada tipo de peza. Seleccionaronse as cores para os dereitos humanos e definironse os temas correspondentes que permiten asociar as cores coas pezas.

Na revisión do *Sprint* validáronse as historias de usuario HU.1 e HU.2. Modificouse a definición da HU.9, establecendo o galego coma idioma predeterminado, e

acordouse que ao recargar a páxina o idioma seleccionado debe de manterse. Con respecto á HU.12, que se abordará no seguinte *Sprint*, acordaronse os parámetros a incluír nos ficheiros de configuración.

5.2. Sprint 2

Neste *Sprint* comezouse coa procura dunha librería que permitira xestionar adecuadamente a internacionalización da aplicación. Realizouse unha breve análise de distintas alternativas existentes en JavaScript, e finalmente optouse por utilizar a librería *jQuery.i18n*. Creáronse os ficheiros cos textos do videoxogo traducidos ao galego e ao castelán, e implementouse o selector de idioma. Por outra parte, estableceuse a arquitectura e a estrutura de directorios definitiva do videoxogo. A partires diso, completáronse as funcionalidades correspondentes ao *Sprint* e fixéronse as probas oportunas.

Na revisión do *Sprint* validáronse as historias HU.9 e HU.12. Ademais dos arquivos de configuración dos niveis, acordouse engadir un arquivo de configuración global para dispoñer de parámetros xerais que poden ser utilizados en distintas partes do videoxogo. Definiuse unha nova historia de usuario para definir o criterio de finalización dun subnivel, e engadiuse ao *product backlog* e ao *sprint backlog* do *Sprint* 3. No seguinte cadro especifícase esta historia:

Título: Criterio de finalización dun subnivel
Identificador: HU.13
Descrición: Como administrador quero poder definir distintos criterios de finalización dos subniveis para poder crear distintos retos para os usuarios e que se esforzen por superar cada subnivel.
Criterio de aceptación: Os criterios de finalización dun subnivel definidos serán dous: ou ben o usuario alcanza un número de liñas predefinido ou ben consegue acadar unha puntuación tamén predefinida para o subnivel en cuestión. Débese poder parametrizar o tipo de criterio seguido e número de liñas/puntos necesarios para superar o subnivel.

Cadro 5.1: Historia de Usuario 13

5.3. Sprint 3

Ao inicio deste *Sprint* realizouse unha procura dunha librería axeitada para substituír as ventás modais de alerta de JavaScript (os *alert()*) para poder construír notificacións máis interactivas e atractivas visualmente para os usuarios. Tras realizar algunhas probas optouse pola utilización de *SweetAlert2*. Engadíronse os textos correspondentes a cada peza, e fíxose a clasificación inicial das pezas que consistiu en establecer a dificultade para encaixar unha peza en función do número de superposicións que pode formar coas pezas que xa están creadas na partida. O motor inclúe por defecto unha función que calcula as superposicións antes de xerar a seguinte peza, e en función do nivel de dificultade xera un tipo de peza ou outro. A partir de aí implementáronse as funcionalidades correspondentes e realizáronse as probas oportunas.

Na revisión do *Sprint* validáronse as historias HU.3, HU.8 e HU.13. Na reunión xurdiu a seguinte historia de usuario, que será incluido nun futuro *Sprint*:

Título: Botón de pausa
Identificador: HU.14
Descrición: Como usuario quero poder pausar unha partida en curso para poder analizar os posibles movementos ou ben tomar un descanso.
Criterio de aceptación: Un usuario que acceda ao videoxogo debe poder pausar unha partida cun botón específico que tamén se activará coa tecla ***p***. Dende o punto de vista do administrador, que apareza ou non o botón en determinados subniveis debe de poder parametrizarse, constituíndo unha dificultade maior naqueles subniveis que non se permita pausar.

Cadro 5.2: Historia de Usuario 14

5.4. Sprint 4

Neste *Sprint* definíronse os dous niveis que en terá o videoxogo. Cada nivel consistirá en catro subniveis, un por cada dereito humano (auga, alimentación, enerxía e TIC) e un nivel mixto final no cal se tratarán todos os dereitos humanos xuntos. No *Sprint* 4 implementáronse os catro subniveis dos dereitos humanos. O nivel mixto final aboradarase nun *Sprint* futuro, dacordo coa planificación. Completaronse o resto de funcionalidades correspondentes ao *Sprint* e executáronse as probas.

Na revisión do *Sprint* validáronse as historias HU.6 e HU.10. Acordáronse algunhas modificacións na parametrización e, a raíz dun novo requisito dos clientes/-titores, xurdiu a seguinte historia de usuario, que se abordará no seguinte *Sprint*:

Título: Textos especiais
Identificador: HU.15
Descrición: Como administrador quero poder amosar textos distintos aos definidos para cada tipo de peza nun momento aleatorio para ter a posibilidade de poder amosar mensaxes e concienciar acerca dalgunhas temáticas distintas dos Dereitos Humanos incluídos. Por exemplo: feminismo, loita contra a guerra e o terrorismo, etc.
Criterio de aceptación: O videoxogo debe de ter a capacidade de poder escoller un texto especial cando se xera unha peza, e asociar ese texto e a cor definida á peza en cuestión. O tipo de texto amosado, a cor e a probablidade de aparición deben de poder parametrizarse nos arquivos de configuración dos niveis. Os textos relacionados co tipo de peza especial débense engadir nos arquivos correspondentes das distintas linguas empregadas no videoxogo, no noso caso galego e castelán

Cadro 5.3: Historia de Usuario 15

5.5. Sprint 5

Comezouse o *Sprint* realizando unha análise dunha librería para poder xerar mapas na aplicación. Tras comparar as distintas alternativas optouse por utilizar *JQVMap*. Esta librería permite xerar mapas vectoriais engadindo o ficheiro JavaScript da librería e o ficheiro CSS correspondente. Cada mapa xerado dispón de numerosas opcións de configuración que permiten modificar a súa apariencia e comportamento.

Por outra parte, analizouse a parte do código do motor que realiza a inicialización do xogo e a animación das pezas, para entender os algoritmos e engadir un novo algoritmo para xerar os obstáculos. Finalmente completáronse e probáronse as funcionalidades correspondentes.

Na revisión do sprint validáronse as historias HU.5, HU.11 e HU.15. Acordouse incluír a HU.14 no *Sprint* 6 ademais de acordar algunhas melloras con respecto á

visualización dos textos asociados ás pezas. Tamén se modificou a clasificación das pezas e dos textos asociados, pasando a definir tres categorías: *easy, normal* e *hard*. Definiuse unha clasificación das pezas en cada categoría, e acordouse parametrizar a probabilidade de aparición de cada tipo nos arquivos de configuración dos niveis. Por outro lado decidiuse almacenar en memoria un rexistro das puntuacións obtidas polo usuario coa idea de poder facer comparativas, cadros de puntuación máximas, etc. Este requisito especifícase na seguinte historia de usuario, que tamén se engadiu ao *sprint backlog* do último *Sprint*:

Título: Rexistro de puntuacións
Identificador: HU.16
Descrición: Como administrador quero poder rexistrar as puntuacións obtidas polo usuario (número de liñas completas e número de puntos obtidos) para poder logo establecer comparativas, facer táboas de puntuacións máximas, etc.
Criterio de aceptación: O videoxogo debe de gardar internamente en memoria o número de liñas e o número de puntos obtidos en cada subnivel. Os totais por nivel e o total global calcularanse coma agregados a partir das puntuacións individuais.

Cadro 5.4: Historia de Usuario 16

5.6. Sprint 6

O *Sprint* final serviu para rematar as historias restantes. Para poder implementar a xeración de pezas con formas personalizadas, realizouse unha análise do algoritmo de xeración de pezas do motor, para conseguir entender como funciona e poder facer as modificacións oportunas para xerar as pezas que se definan, sen afectar á funcionalidade existente e co mesmo comportamento que as pezas que xa xeraba o motor. Completáronse as funcionalidades pendentes e probaronse debidamente. Unha vez feito isto pasouse á fase de finalización co obxectivo de rematar a documentación e

despregar o videoxogo.

Na reunión de revisión do *Sprint*, validáronse as historias de usuario HU.4, HU.7, HU.14 e HU.16.

5.7. Burn down chart

O *burn down chart* ou gráfico de traballo pendente é unha ferramenta de *Scrum* que serve para monitorizar o progreso do desevolvemento do proxecto. Amosa a qué velocidade se están completando as historias de usuario e permite ver se se poderá rematar o traballo pendente no tempo estimado [18].

Na Figura 5.1 podemos ver a relación entre o esforzo planificado (en número de historias de usuario) e o que realmente se levou a cabo en cada *Sprint*. Como se pode observar, a partir do *Sprint* 2 houbo un incremento do traballo real con respeto ao planificado, que se acentuou nos *Sprints* 4 e 5. Isto foi debido a correccións e modificacións dalgunhas historias de usuario de *Sprints* anteriores e tamén pola aparición de novas historias que non estaban contempladas de inicio.

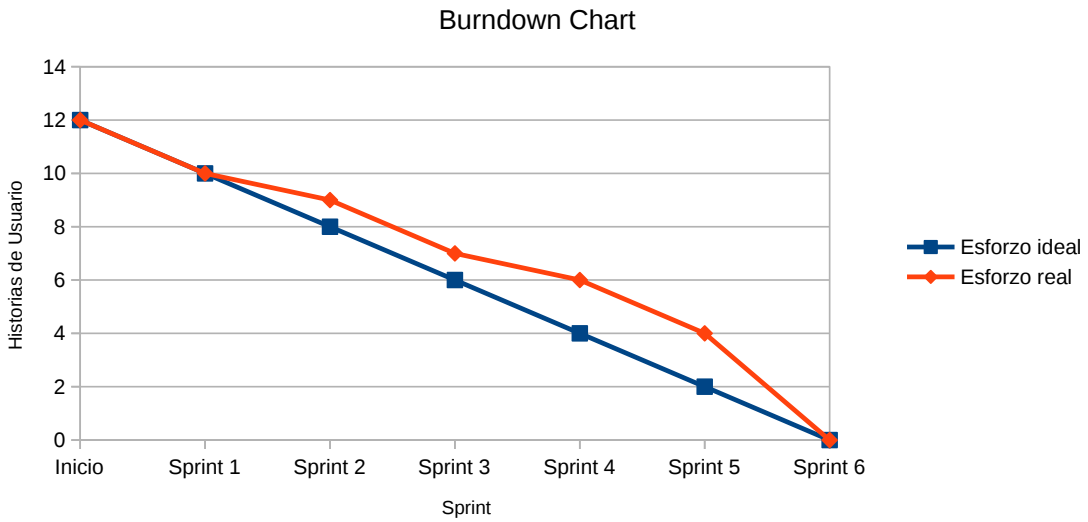

Figura 5.1: Burndown Chart.

Capítulo 6

Deseño e implementación

Neste capítulo describirase o deseño e a implementación do videoxogo. Expónse a arquitectura, amósase a estrutura de clases e descríbense detalles da implementación das funcionalidades máis relevantes.

6.1. Deseño software

O software do videoxogo deseñouse seguindo unha aproximación orientada a obxectos. JavaScript é unha linguaxe o suficientemente flexible como para permitir deseñar e implementar aplicacións seguindo diferentes estilos de deseño e paradigmas de programación, entre eles o paradigma orientado a obxectos. O modelado orientado a obxectos ten como bloques de construción as clases e os obxectos, e é válido para todo tipo de dominios e calquera abanico de tamaños e complexidades.

Estruturar o código adecuadamente permite obter unha aplicación máis sinxela de manter, estender e promove a reutilización de código. Para crear unha estrutura axeitada é fundamental crear modelos que representen a organización interna da aplicación. Á hora de crear ditos modelos emprégase unha notación acordada, coma por exemplo, a linguaxe **UML** (*Unified Modelling Language*). UML é unha familia de notacións gráficas que axudan a describir e deseñar sistemas de software, e en

particular aqueles sistemas construídos cun estilo orientado a obxectos [19]. Baséase na especificación e documentación dos compoñentes dun sistemas a distintos niveis.

Para amosar a estrutura do software do noso videoxogo, empregaremos os catro diagramas que especifica UML:

- **Diagrama de clases**: Describe os tipos de obxectos existentes na aplicación e os tipos de relacións estáticas existentes entre eles. Tamén amosan as propiedades e operacións dunha clase. Empregaremos este diagrama para amosar a estrutura de clases do videoxogo e os patróns de deseño que xurdiron.
- **Diagrama de secuencia**: Describen como colaboran grupos de obxectos nunha determinada interacción. Unha interacción consiste nun comportamento que inclúe mensaxes intercambiados por un conxunto de obxectos dentro dun contexto para acadar un propósito. Utilizaremos este tipo de diagrama para amosar o fluxo de control do videoxogo e o intercambio de información entre os distintos obxectos.
- **Diagrama de compoñentes**: Describen as relacións entre os distintos compoñentes da aplicación. Un compoñente é unha parte física substituíble dun sistema que da soporte a un conxunto de interfaces, representando o encapsulamiento físico dos elementos lóxicos coma clases, interfaces e as relacións entre eles.
- **Diagrama de despregamento**: Describe a estrutura física da aplicación amosando as relacións entre compoñentes hardware e software. Empregaremos este tipo de diagrama en conxunción co diagrama de compoñenetes na Figura 6.2.

Estrutura de clases

De forma global, o videoxogo deseñouse de forma que o motor do xogo fose xenérico, permitindo xerar distintos tipos de xogos baseados nun *tetris*. É dicir,

realizouse un esforzo no deseño para que non sexa específico só para este proxecto, senón que o motor poida ser utilizado noutros xogos de *tetris*. Neste sentido, o código do motor está totalmente desacoplado do resto do videoxogo. Outro aspecto que influíu no deseño foi o esforzo realizado en facer que o maior número de elementos de control do motor e de visualización fosen parametrizables e que se poidan configurar engadindo e modificando os ficheiros de configuración correspondentes.

O videoxogo implementado para este proxecto é o "*Tetris dos dereitos humanos*". Para construílo utilizando o motor xenérico de *tetris*, seguiuse o patrón de deseño **MVP** (*Model - View -Presenter*) [20]. Este patrón permite desacoplar o código que xestiona as interaccións coa interface en tres partes separadas:

- *Model* (Modelo): representa os datos da aplicación e céntrase na súa persistencia.
- *View* (Vista): interface pasiva que contén a lóxica para xestionar os eventos e a visualización dos datos. Comunícase co presentador para actuar sobre os datos de forma que nunca accede ao modelo directamente.
- *Presenter* (Presentador): contén a lóxica para manipular e formatear os datos, cargar ou almacenar información, etc. Actúa tanto sobre o modelo como sobre a vista

Na Figura **6.1** amósase a estrutura de clases do videoxogo.

Como se pode observar, cada unha das partes quedan claramente diferenciadas en tres capas, a de vista, a de presentación e a do modelo. As clases das vistas xestionan a interacción co usuario mediante os eventos que se disparan ao interactuar o usuario coa interface gráfica do videoxogo. Ao recibir un evento, estas clases interactúan coa capa de presentación invocando as funcións correspondentes para xestionar dito evento. Na capa de presentación, extráense os datos do evento e determínanse as accións a executar. Se é necesario, a capa de presentación pode cargar ou modificar datos do modelo. Unha vez actualizado o modelo, devolve a resposta á capa de vista, que se encarga de formatear os datos e actualizar a interface de usuario.

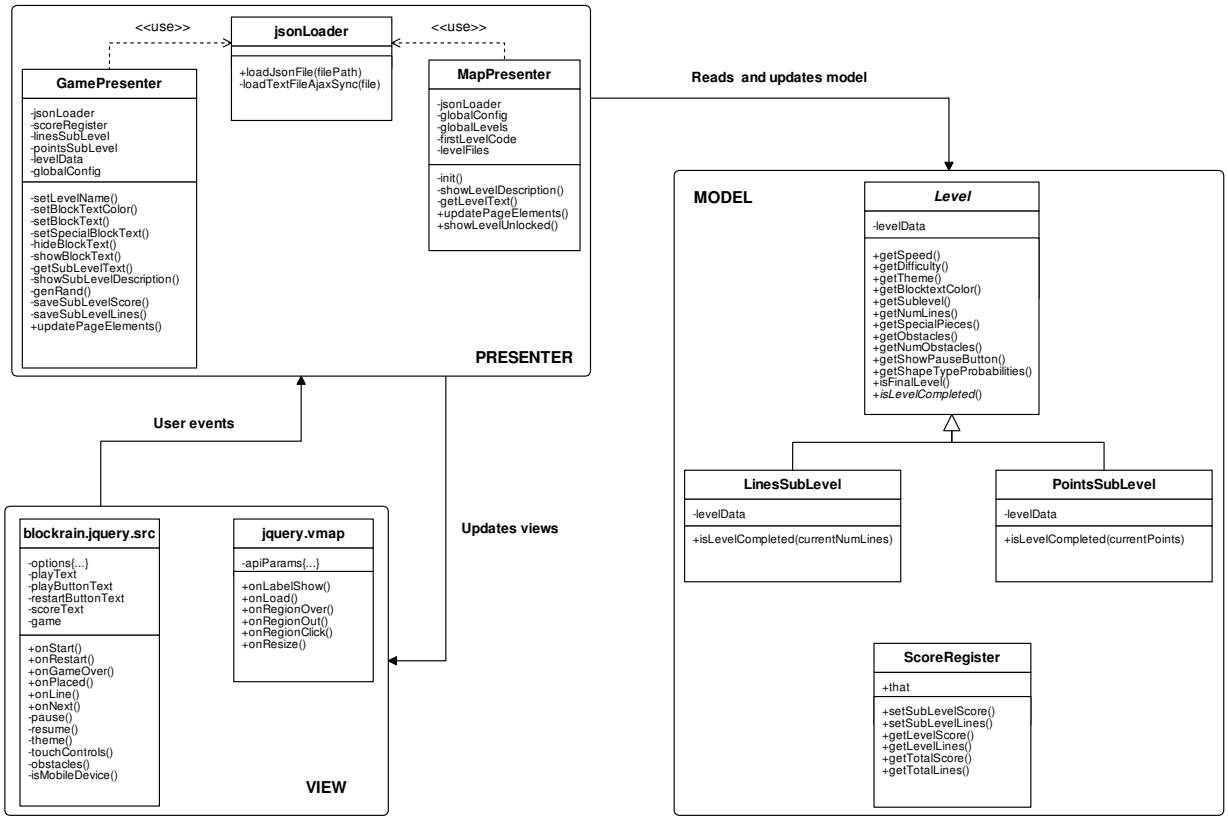

Figura 6.1: Diagrama de clases do videoxogo.

Cabe citar que as clases *GamePresenter, MapPresenter, jsonLoader, Level, ScoreResgister* foron implementados seguindo o patrón de deseño *Module* (módulo) [21]. Un módulo en JavaScript permite encapsular o estado e os atributos do obxecto creando niveis de acceso públicos e privados permitindo emular o concepto de clase existente noutras linguaxes puramente orientadas a obxectos.

6.2. Arquitectura

As aplicacións e videoxogos web empregan unha arquitectura cliente-servidor. Baixo este esquema, o cliente que emprega un usuario para acceder ao videoxogo

é un navegador web, que é de libre elección. O navegador web realiza peticións ao servidor onde está aloxado o videoxogo, quen da a resposta antes as peticións.

No caso do noso proxecto a separación entre os clientes e o servidor será tanto lóxica coma física. Ao final do proxecto o videoxogo será accesible desde os servidores de ESF. Os varios clientes conectados ao servidor executarán cada un a súa propia instancia do videoxogo que se executa no lado do cliente. A arquitectura descrita en detalle amósase na Figura 6.2.

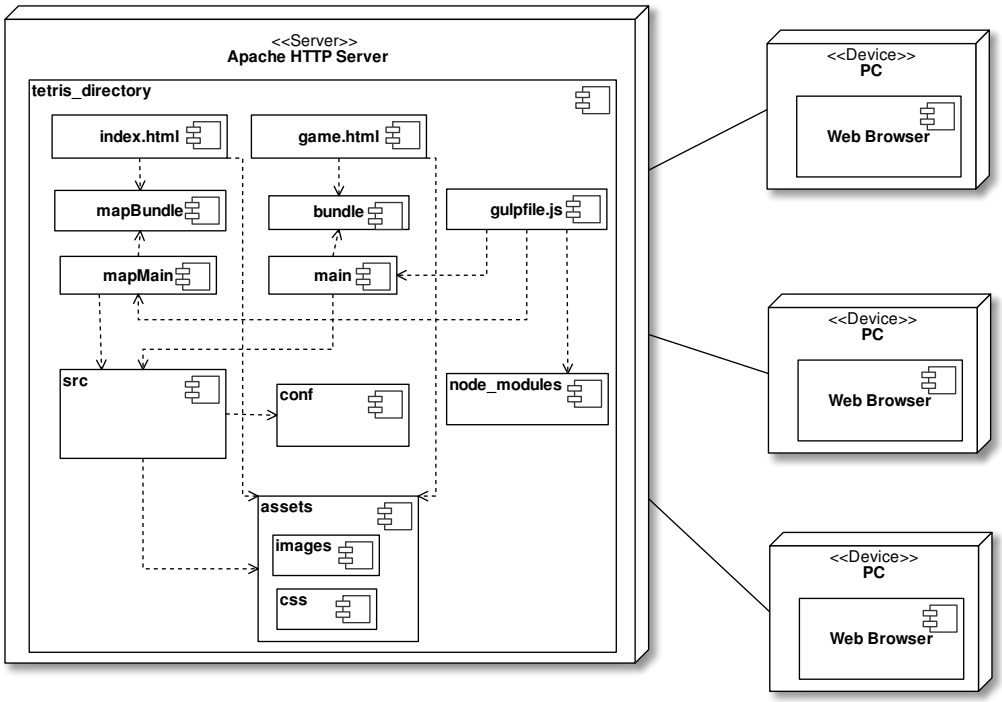

Figura 6.2: Diagrama de arquitectura do videoxogo.

Como se pode observar, o videoxogo estará aloxado nun servidor HTTP Apache, correndo nunha máquina de ESF. Os clientes serán individuais e cada un pode conectarse empregando distintos dispositivos (PCs, móbiles, tabletas..etc).

6.3. Implementación

Nesta sección detállase a implementación das funcionalidades máis relevantes do videoxogo, cunha explicación de como se utilizaron as distintas librerías.

6.3.1. Compilación e automatización

O código do videoxogo debe pasar por un proceso previo de compilación antes de poder ser despregado. Este paso previo é necesario para que os módulos creados poidan funcionar correctamente, sabendo cales son as súas dependencias e como interactuar entre eles.

Como se comentou no apartado 3.3.2, a librería elixida que aporta esta funcionalidade é *Browserify*. A continuación expónse un exemplo de utilización no código do proxecto:

1. No módulo *jsonLoader.js* temos encapsulado o código que permite cargar ficheiros JSON. Para expoñer esta funcionalidade e que poida se utilizada desde outros módulos, engadimos a seguinte función:

```
1 module.exports.loadJsonFile = function loadJson(filePath) {
2   // Load json file;
3   var json = loadTextFileAjaxSync(filePath);
4   // Parse json
5   return JSON.parse(json);
6 };
```

   Coa expresión *module.exports* facemos pública a función *loadJson* para que poida ser invocada dende outros módulos. Acepta como argumento a ruta do ficheiro a cargar.

2. No módulo *GamePresenter*, necesitamos cargar os ficheiros de configuración cos datos dos niveis para poder xestionalos e xerar os niveis cos parámetros

definidos. Para importar a funcionalidade requerida, utilizamos o seguinte código:

```
1 var jsonLoader = require("./jsonLoader.js");
2
3 .
4 .
5 .
6
7 var levelData = jsonLoader.loadJsonFile(levelScript);
```

Para que o código sexa capaz de resolver as dependencias, temos que compilalo co comando *browserify moduleName > bundle.js*. Este comando compila o módulo e xera un *script* de saída denominado *bundle.js*. Este *script* de saída sería o que se debería incluír no noso ficheiro HTML. Por cada módulo creado sería necesario compilalo con este comando e incluír os distintos ficheiros de saída no ficheiro HTML, o cal ocuparía moito tempo.

No código do noso proxecto o proceso de compilación descrito e de xeración do código do videoxogo está automatizado utilizando a librería *Gulp.js*. *Gulp* emprega a función *pipe()* que permite obter todos os datos dun *stream* (fluxo) de entrada e escribilos no destino que indiquemos ou ben envialos como entrada doutro *pipe*. Na seguinte figura vemos un esquema do funcionamento de *Gulp*:

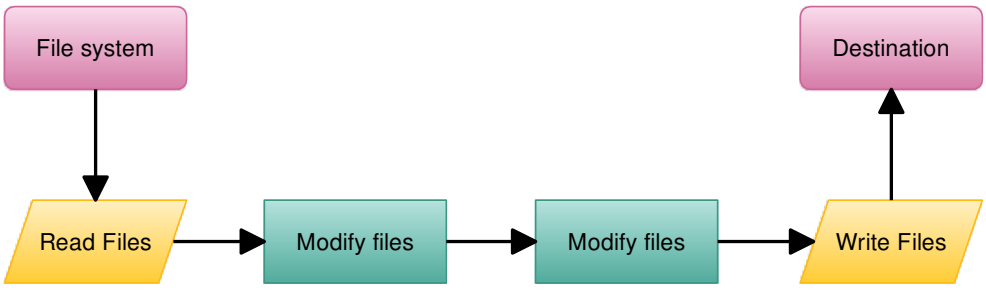

Figura 6.3: Fluxo de funcionamento *Gulp.js*.

O ficheiro de configuración de *Gulp* denomínase habitualmente *gulpfile.js*. Con *Gulp* automatizamos as seguintes tarefas:

- Compilación de módulos: para xerar os módulos compilados, creamos un ficheiro denominado *main.js*, que importa todos os módulos que queremos compilar. Logo tómase este ficheiro como entrada de *browserify*, compílase e xerase o ficheiro de saída *bundle.js* no directorio destino que indiquemos. A continuación vemos os extractos de código de cada unha destas partes:

  - *main.js*:

```
1  var cldrp = require('./src/lib/CLDRPluralRuleParser/
       CLDRPluralRuleParser.js');
2
3  var jqueryi18n = require('./src/lib/jquery.i18n/jquery.
       i18n.js');
4
5  var messagestore = require('./src/lib/jquery.i18n/jquery.
       i18n.messagestore.js');
6
7  var fallbacks = require('./src/lib/jquery.i18n/jquery.i18n
       .fallbacks.js');
8
9  var language = require('./src/lib/jquery.i18n/jquery.i18n.
       language.js');
10
11 var parser = require('./src/lib/jquery.i18n/jquery.i18n.
       parser.js');
12
13 var emitter = require('./src/lib/jquery.i18n/jquery.i18n.
       emitter.js');
14
15 var emitterbidi = require('./src/lib/jquery.i18n/jquery.
       i18n.emitter.bidi.js');
16
17 var history = require('./src/lib/history/jquery.history.js
```

```
    ');
18
19  var url = require('./src/lib/url/url.min.js');
20
21  var languagePicker = require('./src/languagePicker.js');
22
23  var jsonLoader = require('./src/jsonLoader.js');
24
25  var gamePresenter = require('./src/gamePresenter.js');
26
27  var blockrainlibs = require('./src/blockrain.jquery.libs.
    js');
28
29  var blockrain = require('./src/blockrain.jquery.src.js');
30
31  var blockrainthemes = require('./src/blockrain.jquery.
    themes.js');
32
```

- *gulpfile.js*: coa función *gulp.task()* definimos unha tarefa. Esta función toma 3 argumentos: o nome da tarefa, as tarefas que dependen desta e a función executada ao invocar a tarefa. O comando para invocar unha tarefa con *Gulp* é:

```
1   $ gulp taskname
```

A continuación vemos a configuración da tarefa creada para a compilación do código:

```
1   var paths = {
2       scripts: {
3           source: "./main.js",
4           destination: "./",
```

```
5           filename: "bundle.js",
6           watch: "./main.js"
7       }
8   };//Source main file and output dir and filename
9
10  gulp.task("scripts", function() {//browserify task to
        compile scripts. Generates bundle.js
11      var bundle = browserify({
12          entries: [paths.scripts.source],
13          debug: true
14      });
15
16      return bundle
17          .bundle()
18          .pipe(source(paths.scripts.filename))
19          .pipe(gulp.dest(paths.scripts.destination))
20          .pipe(livereload());
21  });
22
```

- Minificar recursos: enténdese coma "minificación" o proceso mediante o cal elimínanse datos innecesarios ou redundantes dun recurso sen que se vexa afectada a forma na cal é procesado [22]. Isto permite reducir o peso dos arquivos de código fonte mediante a eliminación de bytes innecesarios (espazos adicionais, saltos de liña, sangrías ou comentarios), ademais de simplificar a estrutura das sentenzas. Con *Gulp* incorporamos o "minifcado" do HTML e CSS do videoxogo ao fluxo de traballo coas seguintes tarefas:

```
1  var minifyCss = require("gulp-minify-css"),
2      minifyHtml = require("gulp-minify-html");
3
4  gulp.task("html", function() {
5      // Minify HTML
6      gulp
```

```
7           .src("./*.html") // path to your files
8           .pipe(minifyHtml())
9           .pipe(gulp.dest("./dist"));
10  });
11
12  gulp.task("css", function() {
13      // CSS: Concatenate and Minify
14      return gulp
15          .src(["./assets/css/*.css"])
16          .pipe(minifyCss())
17          .pipe(gulp.dest("./dist/assets/css"));
18  });
19
```

- Xeración do directorio de saída: unha vez compilados os módulos e "minificados" os recursos, o que queda é xerar o directorio onde deixar os ficheiros resultantes destes procesos. Crearemos un directorio chamado ***dist*** que terá o código despregable do videoxogo. Para iso, creamos unha tarefa chamada *build* que invoca secuencialmente unha serie de tarefas para eliminar o directorio de saída antiguo, compilar e "minificar" os módulos e o código HTML e CSS, redirixir ao directorio de saída os módulos compilados, ficheiros de configuración, imaxes, etc e finalmente crear o directorio con todos os ficheiros e un *ZIP* con todos os arquivos de código fonte comprimidos. A tarefa é a seguinte:

```
1  gulp.task("build", function(callback) {
2      runSequence(
3          "clean",
4          "html",
5          "scripts",
6          "mapScripts",
7          "bundle",
8          "mapsBundle",
9          "conf",
```

```
        "levelsJSON",
        "i18n",
        "css",
        "blocks",
        "images",
        "readme",
        "dist",
        callback
    );
});
```

Con isto, executado o seguinte comando obtemos o código do videoxogo listo para despregar no directorio *dist*:

```
$ gulp build
```

6.3.2. Internacionalización

O videoxogo debe incluír os textos das distintas pantallas traducidos nos idiomas galego e castelán, permitindo futuras adaptacións a outras linguas, dacordo co descrito na historia de usuario HU.9 do Cadro 3.9.

Para isto, será fundamental a utilización dos códigos de idioma correspondentes. De acordo coa norma ISC 639-1, os códigos son *es* para o castelán e *gl* para o galego. Os usuarios poderán elixir o idioma da súa preferencia cun selector habilitado para tal finalidade. A partir de aí, utilizaremos o código de idioma correspondente para cargar os textos.

Para xestionar internamente a internacionalización e os ficheiros cos textos traducidos, empregamos e librería *jQuery.i18n*. Os ficheiros están en formato JSON, e como convención noméanse incluíndo o código de idioma no que están os textos

do ficheiro. Para interactuar coa librería e invocar funcións empregamos un obxecto *$.i18n()* que inicializa *jQuery.i18n*. Con este obxecto cargamos os ficheiros cos textos (coa función *$.i18n().load()*) e cada vez que o usuario cambia o idioma, modificamos o atributo interno *$.i18n().locale* co código do idioma escollido polo usuario.

Ao mesmo tempo que se actualiza o *locale* interno, actualizamos a URL do videoxogo engadíndolle un parámetro *?locale=gl/es*. Desta forma ao recargar a páxina, mantense o locale previamente seleccionado. Esta funcionalidade implementámola coas librerías *jQuery.history.js* [23] e *js-url* [24].

A continuación amósase un diagrama coa secuencia de control destas accións:

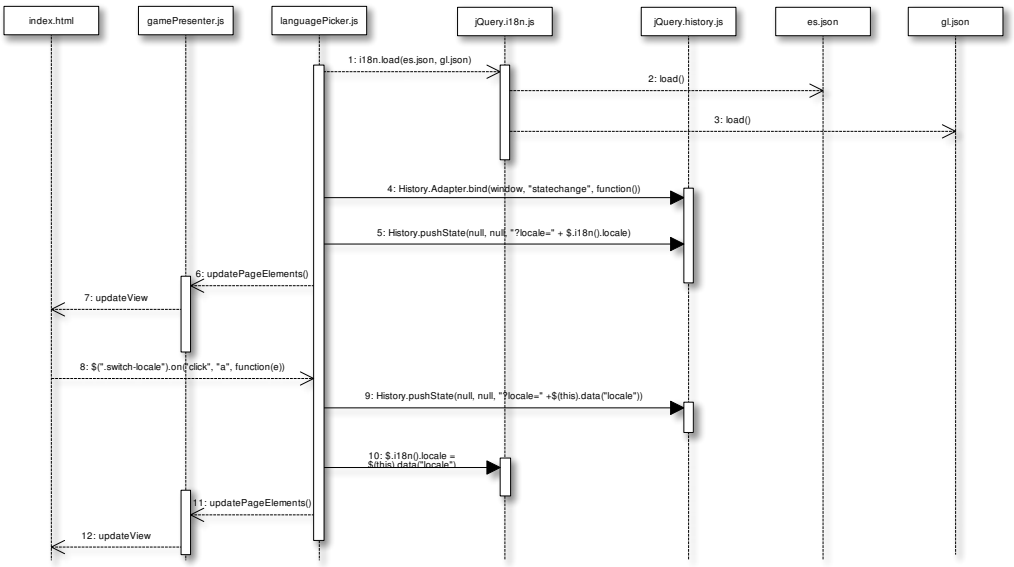

Figura 6.4: Diagrama de secuencia da xestión da internacionalización.

6.3.3. Configuración

Un requisito importante solicitado polos clientes/titores é o de poder parametrizar os distintos niveis e subniveis do videoxogo con arquivos de configuración. Ademais, os arquivos deben de estar enlazados de forma que se carguen os niveis

secuencialmente, simulando unha estrutura dun grafo dirixido, dacordo co descrito na historia HU.12 do cadro 3.12.

Para implementar esta estrutura, creáronse arquivos estruturados en formato JSON coa parametrización dun nivel e os seus subniveis. Implementouse no módulo *jsonLoader.js* a funcionalidade requerida para cargar e procesar estes arquivos en memoria. A estrutura do arquivo de configuración do nivel de España, xunto coa parametrización do seu primeiro subnivel amósase a continuación:

```
{
    "level": "1",
    "next_level_file": "2.json",
    "next_level_code": "hn",
    "levels_params": [{
            "sub_level": "1_1",
            "theme": "water",
            "block_text_color": "#3399ff",
            "speed": 20,
            "difficulty": "normal",
            "shapeTypeProbabilities": [0.45, 0.45, 0.1],
            "obstacles": 0,
            "num_obstacles": 0,
            "num_lines": 2,
            "show_pause_button": true,
            "final_level": false,
            "special_pieces": [{
                "piece_type": "feminism",
                "color": "#8c1aff",
                "piece_probability": 0.4
            }]
        }
    ]
}
```

Os parámetros máis relevantes son os seguintes:

- *next_level_file* e *next_level_code*: referencia o nome do ficheiro do seguinte nivel e o código ISO do país.
- *levels_params*: Un *array* cos distintos parámetros que definimos para un subnivel.
- *sub_level*: Número do subnivel.
- *shapeTypeProbabilities*: expresa as probabilidades de que se xere un tipo de peza. O primeiro parámetro é para as pezas *easy*, que son as máis sinxelas de encaixar. O segundo é para as pezas estándar (*normal*) e o último parámetroé para as pezas *hard*, que inclúe todas aquelas que non son do conxunto de pezas clásico do *tetris*.
- *special_pieces*: *Array* que contén a parametrización das pezas especiais. O parámetro *piece_type* é o nome do tipo de peza e dos textos asociados, o parámetro *color* é a cor na que se renderiza a peza e o texto asociado e por último a *piece_probability* representa a probabilidade de xeración da peza especial en cuestión.

6.3.4. Persistencia

Para algunhas funcións do videoxogo é necesario almacenar algúns datos en memoria de forma persistente. Para iso, necesitamos unha librería que permita almacenar obxectos de forma persistente. Afortunadamente HTML5 conta cunha API de persistencia integrada na súa especificación denominada ***Web Storage*** [25]. *Web Storage* permite almacenar datos nun fomato clave/valor de forma local, no navegador web do cliente.

No videoxogo almacenaremos os datos a nivel de sesión, de forma que cada vez que o usuario acceda de novo terá que empezar desde o inicio do videoxogo. Os datos almacenados son os seguintes:

- Puntuacións: tanto o número de liñas feitas en cada subnivel como a puntuación obtida en cada subnivel.

- Niveis desbloqueados: a medida que o usuario avanza polos niveis, desbloquea os seguintes. Almacenamos o conxunto de niveis desbloqueados.

- Parámetros de control: para manter o estado entre as distintas ventás do xogo, parámetros de control lidos do arquivo de configuración, etc.

A continuación amósase un exemplo de utilización para almacenar a puntuación obtida nun subnivel:

```
1  setSubLevelScore = function(level, subLevel, subLevelScore) {
2          var scores, levelScores;
3          if (sessionStorage["scores"] == null) { // if not exists
       scores, we save it in sessionStorage
4              levelScores = [subLevelScore];
5              scores = {}
6              scores[level] = levelScores;
7              sessionStorage.setItem("scores", JSON.stringify(scores)
       );
8          } else {
9              scores = JSON.parse(sessionStorage.getItem("scores"));
       // reads the saved value
10             if (scores[level] === undefined) { // create new
       element for sublevel score
11                 levelScores = [subLevelScore];
12                 scores[level] = levelScores;
13             } else {
14                 scores[level][subLevel] = subLevelScore; // update
       sublevel score
15             }
16             sessionStorage.setItem("scores", JSON.stringify(scores)
       ); // store the item in sessionStorage
17         }
18     };
```

Capítulo 7

Conclusións e posibles ampliacións

O obxectivo deste proxecto foi o desenvolvemento completo dun videoxogo denominado *"Tetris dos Dereitos Humanos"* para contribuír á difusión de contido sensibilizador no eido dos dereitos humanos. Os aspectos clave do videoxogo serán proprocionar unha interface de usuario adaptable de forma que poida ser xogado en múltiples dispositivos independentemente dos seus sistemas operativos, integrar na dinámica do videoxogo os contidos de dereitos humanos, incluír unha serie de mecanismos sinxelos que permitan engadir novos niveis e modificar o comportamento dos existentes e por último, que todas as ferramentas e tecnoloxías empregadas no seu desenvolvemento sexan software libre.

O proxecto comezou de forma ambiciosa coa proposición por parte dos clientes/-titores dunha ampla serie de funcionalidades e obxectivos. Como as funcionalidades iniciais non estaban completamente especificadas, e era probable que xurdisen novas funcionalidades e requisitos ao longo da vida do proxecto, decidiuse seguir unha metodoloxías de desenvolvemento áxil para poder ter unha mellor resposta aos cambios. Ao longo dos *Sprints* os requisitos fóronse refinando e adaptando ás necesidades dos clientes/titores, e tamén apareceron novos requisitos que non estaban contemplados inicialmente. Grazas ao tipo de metodoloxía escollida estes cambios non supuxeron un problema inabarcable e púidose cumprir cos requisitos fundamentais para acadar

os obxectivos do proxecto.

Destaca a estrutura interna do motor do videoxogo, o cal está desacoplado do código que xera o videoxogo *"Tetris dos Dereitos Humanos"*. Está implementado coma un plugin de *jQuery* o cal facilita a súa utilización noutros tipos de *tetris*. Tamén é importante destacar o mecanismo implementado para parametrizar o videoxogo, o cal permite engadir novos niveis/subniveis de forma rápida e sinxela engadindo o arquivo de configuración correspondente seguindo a estrutura definida.

No que atinxe ás tecnoloxías escollidas para a implementación, cabe citar o acerto que se tivo neste sentido. Grazas ás funcionalidades que inclúe a librería estándar de JavaScript e a gran cantidade de librerías de terceiros dispoñibles non se tivo ningún problema en desenvolver ningunha das funcionalidades. Foi importante a elección de tomar como base o proxecto libre *blockrain.js*, que proporcionou unha base axeitada sobre a cal se construíron o resto de funcionalidades. A análise inicial e o desenvolvemento dun prototipo inicial empregando *blockrain.js* foron claves neste sentido. A utilización de tecnoloxías web tamén resultou acertada á hora de implementar o videoxogo de forma que poida ser xogado dese distintos dispositivos con tamaños de pantalla distintos. O sistema operativo do dispositivo non debe de ser un problema xa que só se require un navegador web compatible.

Finalmente, cabe destacar que a realización deste proxecto serviu para aplicar algunhas das competencias adquiridas no mestrado tanto a nivel de xestión de proxectos, de análise e deseño, de planificación temporal ou de interacción web con HTML5.
Ao mesmo tempo tamén se ampliaron competencias tanto a nivel tecnolóxico, por empregar algunhas tecnoloxías das cales non se tiña coñecemento previo coma a nivel de xestión e planificación, mellorando as capacidades de estimación e aprendendo a aplicar unha metodoloxía de desenvolvemento áxil.

7.1. Ampliacións e melloras

Expóñense a continuación as posibles melloras do videoxogo existente e as ampliacións futuras:

- Persistencia en base de datos: os totais das liñas feitas e a puntuación total obtida pódense almacenar nunha base de datos coma *SQLite* a nivel de usuario. Desta forma pódense crear listas de puntuacións máximas e fomentar o pique sano entre os usuarios.
- Preguntas entre niveis: como proba final para superar un nivel, poden aparecer un par de preguntas ao final sobre os dereitos humanos tratados para comprobar a asimilación dos conceptos por parte dos usuarios, enagdindo un maior reto ao videoxogo e dándolle unha maior importancia aos contidos de dereitos humanos.
- Logros: contémplase a posibilidade de engadir logros a modo de recompensa para os usuarios que acaden determinados obxectivos. Pode servir para aumentar a motivación e a competitividade dos usuarios.
- Engadir máis niveis: finalmente incluíronse dous niveis, España e Honduras, cada un cos seus catro subniveis e nivel final mixto. Sempre se poden engadir máis niveis para tratar realidades distintas e ter unha visión máis ampla do estado dos dereitos humanos en distintos países do mundo.

Apéndice A

Manual de Usuario

Neste apéndice explícase como acceder e interactuar co videoxogo desde o punto de vista dun usuario. Expóñense os distintos navegadores web que se poden utilizar e amósanse exemplos de interacción tanto desde un PC coma desde un dispositivo móbil.

A.1. Cliente Web

Para poder interactuar co videoxogo será imprescindible ter instalado un navegador web. É importante asegurarse de ter habilitada a execución de *JavaScript* para que o videoxogo funcione correctamente. Do contrario, será imposible interactuar con el. Existen múltiples navegadores web tanto para computadores de escritorio como para dispositivos móbiles. A continuación amósase o listado dos navegadores compatibles:

- **Microsoft Edge (Windows 10, Windows 10 Mobile, iOS, Android 4.4+)**: navegador web desenvolvido por Microsoft, é o reemplazo de Internet Explorer. Está construido en torno a estándares web e inclúe un novo motor de renderizado, *EdgeHTML* o cal está baseado en *Trident*.

- **Google Chrome (Windows 7 ou posterior, Mac OS X 10.9 ou posterior, Linux, Android 4.1+, iOS 9+)**: navegador web desenvolvido por Google. Dispoñible gratuitamente, ten un proxecto de software libre chamado *Chromium*.
- **Mozilla Firefox (Windows 7 ou posterior, Mac OS X 10.5 ou posterior, Linux, Android 4.1+, iOS, Firefox OS)**: navegador web libre e de código aberto desenvolvido pola Fundación Mozilla. Utiliza o motor de renderizado *Gecko* o cal implementa actuales e futuros estándares web.
- **Opera (Windows 7 ou posterior, Mac OS X 10.10 ou posterior, Linux, Android, iOS, Windows Mobile)**: navegador web da empresa Opera Software. Utiliza o motor de renderizado *Blink*. Está dispoñible gratuitamente.
- **Safari (macOS, iOS)**: navegador web de código pechado desenvolvido por Apple Inc. Dispoñible para os seus sistemas operativos, macOS e iOS.

A.2. Interacción co videoxogo

Unha vez instalado o navegador web da nosa preferencia, para acceder ao videoxogo emprégase a seguinte URL:

`https://camin246.udc.es/tetris/`

Amosamos a continuación unha serie de pantallazos da interacción co videoxogo, utilizando o navegador web Mozilla Firefox:

- Páxina inicial:

Figura A.1: Páxina de inicio do videoxogo.

- Selección nivel España:

Figura A.2: Nivel de España seleccionado.

- Nivel España (Dereito Humano Auga):

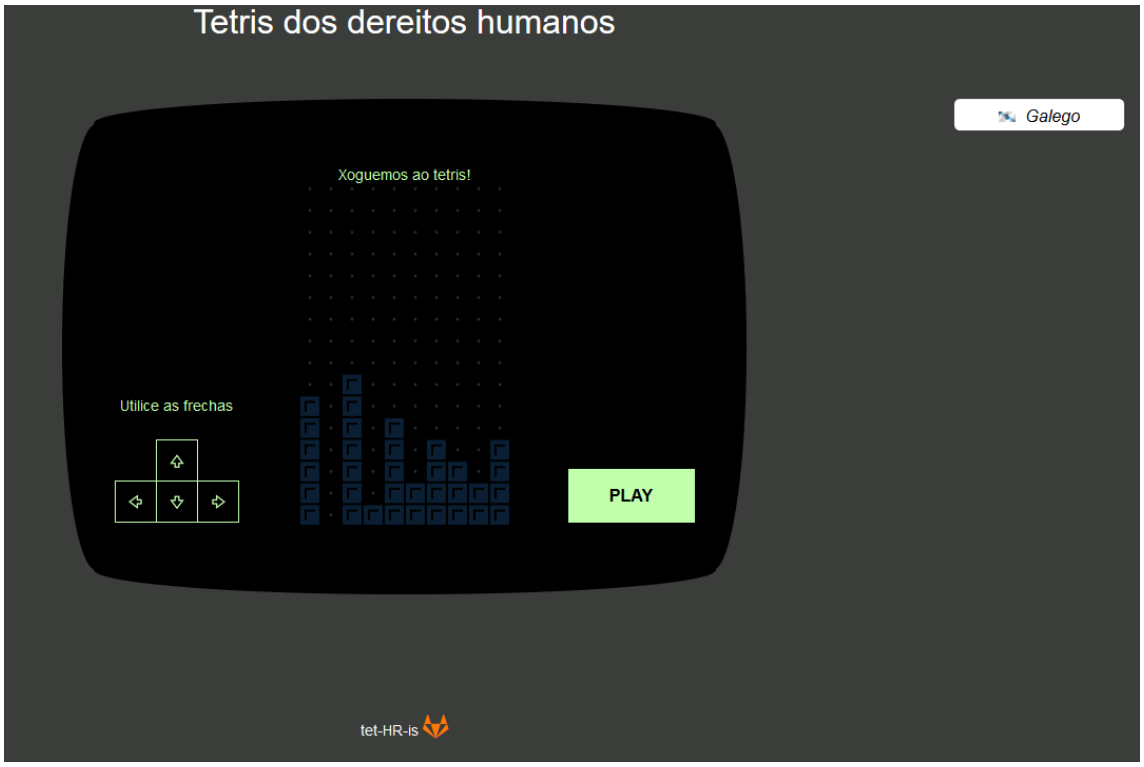

Figura A.3: Inicio do nivel de España.

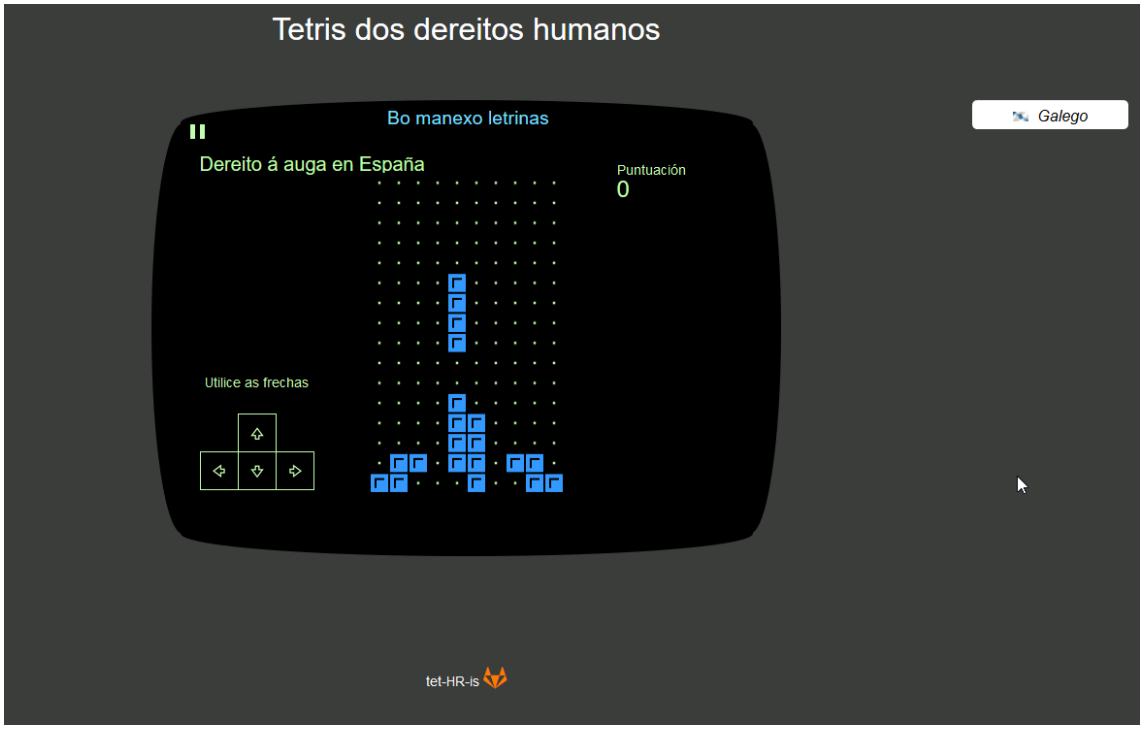

Figura A.4: Nivel de España (Dereito Humano Auga).

Bibliografía

[1] Definición de Software Libre. Free Software Foundation. *https://www.gnu.org/philosophy/free-sw.gl.html* → páxs 3, 26

[2] Julio Salas Bacalla, Oscar Tinoco Gómez e Pedro Pablo Rosales López, *Criterios de selección de metodologías de desarrollo de software*, Industrial Data, vol. 13, núm. 2, julio 2010, pp. 70-74. → páx. 5

[3] Principles behind the agile manifesto. *http://agilemanifesto.org/principles.html* → páx. 6

[4] Jeff Sutherland e Ken Schwaber, *The Scrum Guide*, Scrum Alliance, 2011. → páx. 7

[5] ScrumBut. *https://www.scrum.org/ScrumBut* → páx. 7

[6] Roger S. Pressman, *Ingeniería del software, un enfoque práctico*, 7ª edición, McGrawHill. → páx. 11

[7] Documentación de Git. *https://git-scm.com/documentation* → páx. 12

[8] Project Management Institute, *Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK)*, 3ª Edición. → páx. 13

[9] Guía Salarial Sector TI Galicia 2015-2016. *https://es.scribd.com/document/288511179/Guia-Salarial-Sector-TI-Galicia-2015-2016* → páx. 14

[10] Precio del kWh en 2018. Tarifas de gas y luz. *https://tarifasgasluz.com/faq/precio-kwh-espana-2018#precio-kwh-luz-2018* → páx. 15

[11] Electricity usage of a Laptop or Notebook. Energy Use Calculator. *http://energyusecalculator.com/electricity_laptop.htm* → páx. 15

[12] Artículo sobre Gulp.js. *https://frontendlabs.io/1669–gulp-js-en-espanol-tutorial-basico-primeros-pasos-y-ejemplos* → páx. 28

[13] Documentación de uso de Browserify. *https://github.com/browserify/browserify#usage* → páx. 29

[14] Guía avanzada de jQuery.i18n. *https://phraseapp.com/blog/posts/jquery-i18n-the-advanced-guide/* → páx. 29

[15] Páxina oficial de SweetAlert2. *https://sweetalert2.github.io/* → páx. 29

[16] Guía de utilización de XMLHttpRequest. *https://developer.mozilla.org/es/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest* → páx. 30

[17] Documentación JQVMaps. *https://www.10bestdesign.com/jqvmap/documentation/* → páx. 30

[18] Gráficos de trabajo pendiente. *https://proyectosagiles.org/graficos-trabajo-pendiente-burndown-charts/* → páx. 47

[19] Martin Fowler, *UML Distilled*, 3ª edición, Addison-Weasley. → páx. 50

[20] Comparativa entre MVC e MVP. *https://www.infragistics.com/community/blogs/b/todd_snyder or-mvp-pattern-whats-the-difference* → páx. 51

[21] Addy Osmani, *https://addyosmani.com/resources/essentialjsdesignpatterns/book/#modulepatte* O'Reilly. → páx. 52

[22] Minificación de recursos. *https://developers.google.com/speed/docs/insights/MinifyResources?hl= 419* → páx. 58

[23] History.js. *https://github.com/browserstate/history.js* → páx. 61

[24] JS-URL. *https://github.com/websanova/js-url* → páx. 61

[25] HTML5 Web Storage. *https://www.w3schools.com/html/html5_webstorage.asp* → páx. 63