PRÁCTICA 5

Prototipo tecnológico

The Video Game Box

INTEGRANTES DEL GRUPO:
MARIO CAMPOS SOBRINO
CARLOS CARNERO MÉRIDA
JOSÉ DÍAZ REVIEJO
DAVID ELÍAS PIÑEIRO
CARLOS GÓMEZ LÓPEZ
ÁLVARO GÓMEZ SITTIMA
JULIÁN MOFFATT
JUAN ROMO IRIBARREN
JAVIER DE VICENTE VÁZQUEZ
GONZALO VÍLCHEZ RODRÍGUEZ

0. Índice

1. Introducción

El proceso de desarrollo de esta práctica comenzó el martes, cuando nos reunimos todo el equipo en el laboratorio y revisamos los errores cometidos en la anterior práctica en cuanto a la realización del *Product Backlog*. De la corrección y mejora del *Product Backlog* (cuyos cambios se detallan más adelante) se han encargado Julián (*Product Owner*) y José (*Scrum Máster*).

Tras dar por finalizada la revisión y también en el laboratorio, todos los miembros del equipo de desarrollo procedimos a analizar el posible modelo de la arquitectura que utilizaríamos.

Una vez definida la arquitectura a utilizar, lo siguiente fue decidir entre todo el equipo de desarrollo la funcionalidad a implementar en el prototipo, en el que la idea principal es tratar de probar y familiarizarse con el uso de las diversas herramientas y metodologías que usaremos en nuestra aplicación final. Lo siguiente fue repartir y planificar el trabajo para llevar a cabo la implementación de este. Los detalles del prototipo realizado y como se ha llevado a cabo se desarrollan más adelante.

En cuanto a la realización de la memoria, Javier fue el encargado de documentar las decisiones de diseño tomadas y el modelo de la arquitectura utilizada. En relación con el prototipo, Carlos C. se ha encargado de redactar la *Configuración del entorno de administración de la configuración*, Gonzalo de reflejar las *Herramientas y tecnologías probadas en el prototipo*, Carlos G. de explicar todo el proceso de implementación y Mario de indicar cómo ejecutar y probar el prototipo. Los encargados de explicar la corrección y mejora de la primera versión del Product Backlog han sido Julián y José.

Se ha cumplido el objetivo de probar las tecnologías y arquitecturas que más adelante se utilizarán, a pesar de no haberlas podido implementar al 100%, pero no supone un problema pues estamos en una metodología ágil, cumpliendo el hecho de haber desarrollado un ejecutable funcional.

Para finalizar, David fue el responsable de realizar esta introducción basándose en las anotaciones previas sobre cómo fuimos llevando a cabo la práctica y lo que llegó a realizar cada miembro del equipo.

2. Arquitectura

Con respecto a la arquitectura, hemos optado por una *arquitectura multicapa* dividida en dos capas: *negocio* y *presentación*. Aunque comúnmente se suele dividir en 3 capas, al hacer uso de *Java Persistence API* (JPA), solo son necesarias 2 de ellas, ya que la tercera (integración), lo gestiona JPA. La primera será la responsable de la comunicación con los sistemas externos. La de negocio se encargará de proporcionar los servicios al sistema, implementando una clase intermedia *Function*, que hace de intermediario entre las vistas y la base de

datos, con sus respectivos transfer y la entidad *Person*. En esta capa también se encontrará el *Entity Manager*, y por último la capa de presentación, organizada en un paquete donde cada operación *CRUD* (Crear, leer, actualizar, borrar) tiene su propia vista, además de una vista principal y sus respectivos paneles; todo ello implementado con Swing. Hemos utilizado los patrones justos y necesarios para simplificar el código como son el Singleton, restringiendo la creación de objetos a una clase; transfer, que facilita la comunicación de componentes entre capas. Puesto que es un prototipo tecnológico tampoco es necesario indagar en el uso de patrones, aunque para el proyecto final si se tendrá en cuenta el uso de una serie de patrones que mejoren la calidad del software, ya que no hay que reducirla nunca.

Respecto a la base de datos, hemos acordado tras bastante investigación de recursos y muchas pruebas con *MongoDB*, decidimos utilizar para este prototipo tecnológico una base de datos relacional, primando el hecho de conseguir un ejecutable funcional al de abordar tecnologías complejas que más adelante podremos aprender a usar mejor, sobre todo de cara a los sprints.

En cuanto a la conexión de la base de datos, en un principio se apostó por utilizar *Hibernate*, ya que permite acceder de una manera consiste a la capa de persistencia, pero al no estar familiarizados con su uso y a pesar de pelearnos bastante no fuimos capaces de implementarlo, aunque la idea es que para la entrega final si sea implementado. Es por ello por lo que al final se ha utilizado una base de datos relacional, concretamente *MySQL*, que todos conocemos. Además, gracias a haber utilizado JPA, ha sido muy viable la interacción entre la base de datos, persistiendo la entidad de manera asociada a una tabla.

Con un documento descriptivo *.xml* hemos especificado las diversas dependencias, tanto para el uso de JPA como de MySQL, así como de MariaDB. Esto se encuentra en el fichero pom.xml. Al ser un proyecto Maven ha facilitado esta característica, que añade todas las librerías de manera automática.

3. Prototipo

Esta práctica se centra en realizar un prototipo tecnológico, que no trata sobre nuestra aplicación, es decir que tampoco tendrá ninguna de las funcionalidades de ésta, si no que se trata de un prototipo que nos va a servir para familiarizarnos con las herramientas y tecnologías que se van a utilizar en la aplicación. En este apartado se van a explicar las herramientas y tecnologías probadas, cómo se llevo a cabo el desarrollo del prototipo y el cómo ejecutarlo, y por lo que se va a empezar a explicar que es el entorno de administración de la configuración.

3.1. Configuración del entorno de administración de la configuración

Trabajando en GitHub hemos creado un nuevo repositorio al que hemos añadido 5 directorios: código, diseño, documentación, recursos y reuniones.

Como de momento algunos de ellos no tienen contenido, hemos tenido que añadir un directorio *.gitkeep* ya que GitHub no permite añadir directorios vacíos.

El directorio *código* está pensado para desarrollar todo el código de los proyectos, que de momento está dividido en *prototipo* y la carpeta del proyecto, *TheVideoGameBox*. Iremos añadiendo proyectos y cada proyecto estará dividido en presentación, negocio y pruebas.

El directorio *diseño* está pensado para incluir todos los diagramas del proyecto. De momento hemos incluido los directorios *Diagramas de actividad*, *Diagramas de clase* y *Modelo de dominio*.

El directorio *documentación* contendrá todos los documentos para informar del proceso Scrum. Incluye los directorios *Burndown*, *Memorias*, *Impediment Backlog*, *Product Backlog*, *Sprint Backlog* y *Story Map*.

El directorio *recursos* contendrá los elementos disponibles para el correcto funcionamiento del proyecto (imágenes, logos, instaladores de herramientas que utilicemos, controladores, etc.).

El directorio *reuniones* incluirá las grabaciones de vídeo y conclusiones de las reuniones realizadas a lo largo del proyecto (*Daily Scrum*, *Reunión de revisión*, *Reunión de planificación*, *Reunión de retrospectiva*), a partir del inicio del primer sprint.

3.2. Herramientas y tecnologías

A continuación, comentamos las herramientas y tecnologías que hemos probado para realizar este prototipo, y con cuales nos hemos decantado para utilizar.

Empezamos explicando el sistema de control de versiones que hemos probado. Hemos decidido utilizar *GitHub*, que es una plataforma de desarrollo colaborativo que nos permite gestionar el proyecto y las versiones del código de forma gratuita. GitHub está basado en *Git*, que es un sistema de control de versiones que hemos practicado en la asignatura y que conocemos medianamente. Esta herramienta es la que más nos ha gustado comparado con almacenar nuestro proyecto en una nube tipo *Google Drive*, puesto que *GitHub* está especializado en proyectos de software y es la plataforma más utilizada por desarrolladores.

Con respecto al entorno de programación, hemos probado varios, como *IntelliJ*, *Xcode* o *Eclipse*. Mientras que todos estos nos permiten utilizar directamente *GitHub* desde la aplicación, nos hemos decidido al final por *Eclipse*, ya que la totalidad de los integrantes del grupo ya lo han utilizado y lo conocen. Mientras que *Xcode* e *IntelliJ* son aplicaciones comerciales, *Eclipse* está basado en software libre, algo que nos ha parecido mejor utilizar en un entorno educativo. Además, *Xcode* solo está disponible para usuarios de *MacOS*, por lo que hemos decidido usar un IDE multiplataforma como Eclipse que puedan usar todos los integrantes del equipo de desarrollo independientemente del sistema operativo de su ordenador. *Eclipse* además cuenta con numerosas extensiones que pueden sernos útiles en el futuro.

En cuanto a la tecnología de la aplicación, hemos utilizado el lenguaje de programación *Java en su versión 1.17*, ya que es el lenguaje que más dominamos y que ya hemos usado en otras asignaturas. La versión escogida se debe a que es la última versión de *Java* disponible, lo cual nos va a dar acceso a funcionalidades que contenían versiones anteriores, como a nuevas que se han añadido. Por otro lado, es ampliamente usado a nivel profesional, aunque en los últimos años está pasando a un segundo lugar.

También vamos a usar *Maven* que es una utilidad para Java que descarga de forma automática librerías externas y drivers, lo cual nos facilita el trabajo en gran medida ya que no tendremos que buscarlos de forma individual y tratar de encontrar versiones compatibles ya que gracias a esta utilidad este proceso será automático.

Para el desarrollo del prototipo hemos probado tanto *MongoDB* como Hibernate con una base de datos relacional SQL. Las ventajas de MongoDB como base de datos no relacional son que son mucho más flexibles a la hora de almacenar información, sobre todo con datos no estructurados o que casi nadie del equipo las ha utilizado por lo que sería una oportunidad para aprender sobre una tecnología cada vez más utilizada. Este tipo de bases de datos sin embargo no es compatible con cierto tipo de consultas SQL que podrían sernos útiles. Por otro lado, hemos probado a utilizar una *base de datos relacional*, cuyo funcionamiento es ampliamente conocido por todos los miembros del equipo, por lo que sería más fácil desarrollar algunas ideas más complejas, además de que utilizan el lenguaje SQL con el que estamos todos familiarizados. También hemos probado *Hibernate y JPA* como puentes entre la base de datos y nuestra aplicación, que presenta ciertas ventajas, como que ambos son de software libre y simplifican la codificación de consultas a la base de datos y la transformación entre objetos de la aplicación y registros de una tabla. Además, estas librerías funcionan con distintos tipos de bases de datos por lo que sería más fácil intercambiarlas sin cambiar el código. Hibernate es también una herramienta que es mayoritariamente desconocida por los miembros del equipo, mientras que es ampliamente utilizada a nivel profesional, por lo que puede ser una buena forma de adquirir experiencia con ella. Al final, nos acabamos utilizando una base de datos SQL (MySQL MariaDB) junto con JPA, ya que con MongoDB e Hibernate no fuimos capaces de conseguir que funcionase y preferimos utilizar SQL para conseguir un prototipo funcional cumpliendo la fecha de entrega. Gracias a JPA no será complicado cambiar la base de datos a una no relacional más tarde.

Hemos utilizado *MongoDB Compass* como herramienta para diseñar la base de datos de MongoDB puesto que permite explorar la estructura de documentos de forma gráfica e intuitiva. Esta herramienta es además multiplataforma y gratuita para los miembros de nuestro equipo. Para hostear la base de datos hemos probado *MongoDB Atlas*, que es un servicio gratuito a nivel básico al nivel que lo utilizaremos. Por cuestiones de tiempo y fallos que nos hemos encontrado acabamos utilizando SQL y por lo tanto utilizamos MySQLWorkbench que es una herramienta visual de diseño de bases de datos MySQL que hemos utilizado previamente y tenemos bastante experiencia utilizándola.

Para almacenar la base de datos de forma compartida hemos decidido utilizar Azure Database. Esta nube es de las más utilizadas por grandes aplicaciones y garantiza la seguridad en el almacenamiento de datos y la fiabilidad de las conexiones. Además, los servicios de la universidad proporcionan créditos para poder gastarlos en cualquier servicio de Azure.

3.3. Desarrollo del prototipo

A la hora de desarrollar el prototipo, lo primero que se realizó fue una separación de trabajos dentro de éste. *Carlos G., Carlos C. y Mario* se encargaron de realizar la *capa de presentación* del prototipo, repartiéndose entre ellos las vistas a realizar y siguiendo un mismo formato declarado previamente, para así no tener problemas luego de tener que cambiarlo.

A la hora de seguir un curso, se empezó diseñando una vista principal, la cual iba a contener todo lo que se iba a implementar en el prototipo, desde todos los botones para acceder a las distintas funcionalidades del *CRUD*, como el diseño para que se viesen todas las entidades. Tras realizar esta vista todos juntos, se repartieron el resto de las funcionalidades a realizar.

Por otro lado, en donde más integrantes del grupo estuvieron ayudando fue en configurar la base de datos de *MongoDB*, ya que al ser una nueva herramienta para todos se tuvo que investigar mucho sobre ella y como poder implementarla con Hibernate. *Javier, Juan, Álvaro, Gonzalo y David* se encargaron de ello, donde tuvimos la suerte de que Gonzalo y David ya habían llegado a trabajar algo con esta herramienta y que Javier obtuvo mucha información que ayudó a ponerla en funcionamiento. Pero finalmente acabamos usando una base de datos relacional, almacenada en *Azure Database*, ya que no conseguimos conectar Hibernate con MongoDB.

En general, la mayor dificultad que tuvimos fue el tema de *conectar Hibernate con MongoDB*, ya que al usar Hibernate un archivo *persistance.xml*, a la hora de conectarnos, salieron problemas que no fuimos capaces de solventar por mucho que buscamos, lo cual nos llevó a en este prototipo tirar la toalla y recurrir a lo que ya conocíamos, *MySQL*. Para un futuro hemos decidido investigar más, para intentar conectarla y así tener más comodidad a la hora de introducir información de las APIs.

Otra dificultad que tuvimos fue el *configurar Hibernate*, ya que solo dos de los integrantes lo habían usado previamente. Esto consumió más tiempo del esperado, lo cual nos retrasó a la hora de codificar. Se solucionó básicamente poniéndole tiempo e investigando por internet el cómo poder solventarlo.

Finalmente, una última dificultad fue la *configuración de la base de datos*, ya que al ser la primera vez que se usaba, había mucho desconocimiento de ésta, y luego también mucho que codificar dentro de ella. Su solución no varía con respecto a la anterior, se resolvió con tiempo e investigación.

3.4. Instrucciones de ejecución del prototipo

Una vez completado el prototipo, lo exportamos como ejecutable. Para ejecutar el prototipo se deben seguir una serie de pasos y restricciones. Una de las restricciones para poder ejecutarlo es tener la versión **JAVA 1.17**, si no se tiene dicha versión saldrá un error y no se podrá ejecutar.

Tras haber informado de la versión necesaria, para ejecutar el prototipo, deberás acceder al archivo *PrototipoTheVideoGameBox.jar* que se encuentra dentro de la carpeta de entrega.

Una vez ejecutado el jar, deberás seguir las siguientes instrucciones para probar el prototipo:

1. Al entrar en la aplicación, se desplegará la pantalla inicial, en la cual veremos el listado de personas ya guardadas. Todas las personas aparecerán en cuadros cos dos botones (editar y eliminar) y el nombre encima de ellos.
2. A su vez encima de la lista de personas, veremos un botón con la opción de *Añadir Persona*. Al darle, acto seguido se desplegará una nueva ventana *Alta Persona* donde se nos pedirá el NIF(seguir criterio de 8 números y una letra, recomendado poner *98765432X*), el nombre y los apellidos (obligatorios de rellenar). Si incumplimos con estos requisitos, al darle al botón de *Submit* saldrá una ventana mostrando el error. Añadimos la información y damos al botón de *Submit*. Se comunica con las bases de datos y crea el alta de la persona.
3. Nos redirige automáticamente a la pestaña principal y veremos un cuadro nuevo con esa persona. Al hacer click en el nombre de dicha persona nos aparecerá una nueva ventana *Buscar Persona* donde se mostrará toda la información de la persona indicada. Para volver a la pestaña principal presiona el botón *Volver al Principio*.
4. Si deseamos cambiar o editar su información podemos pulsar el botón de *Editar*. Se abrirá una pestaña nueva donde pondremos los nuevos datos, en caso de que alguno se quiera mantener, automáticamente si pulsas su panel, se guardará con la información anterior (recomendable solo cambiar el nombre). Si no rellenamos todos los campos necesarios se mostrará un mensaje informando el error. Una vez editada la persona, damos al botón de *Submit* que comunicará con la base de datos y actualizará las tablas.
5. Nos redirige automáticamente a la ventana principal donde en caso de que se quieran ver los cambios realizados, realizaremos la misma acción que en el *paso 3*.
6. Para dar de baja una persona, tendrás que dirigirte a la persona que quieras eliminar y pulsar el botón *Eliminar* que abrirá una nueva ventana. Esta nueva pestaña nos enseñará la información de la persona a eliminar y nos ofrecerá eliminarla o no. Pulsando el *No*, se nos redirige a la ventana principal sin hacer cambios. Pulsando el *Confirmo*, se comunica con la base de datos y elimina la persona de las tablas. Automáticamente realizando cualquiera de las dos opciones, se cerrará la ventana y si se

ha llegado a confirmar su baja, se podrá ver reflejado en la lista de la ventana principal.

4. Product Backlog

A continuación, nos vamos a centrar en el *Product Backlog*, el cual a partir de las correcciones obtenidas en la práctica anterior y al ser un documento que cambia con el tiempo, en este apartado se van a explicar todos los cambios realizados por parte del *Product Owner* (Julián) y *Scrum Master* (José).

Empezamos explicando las *Historias de Usuario* (HU) que se han agrupado, las cuales son las que mostraban los resultados de una determinada búsqueda dentro de la búsqueda correspondiente a la que hacía referencia. La razón por la que se han unido es porque están tan ligadas que forman parte de la misma acción y no tiene sentido separarlas.

También se han revisado y retocado las descripciones que mencionan detalles técnicos, los cuales no deben estar en la descripción. Al revisar las descripciones hemos intentado que todas sigan la plantilla estudiada en clase dado que en algunas ocasiones el beneficio no estaba bien definido o era algo ambiguo.

Para crear los *identificadores* (IDs) de las *HU* hemos empezado poniendo las siglas del proyecto (**TVGB**), para que no se confundan las *HU* de este proyecto con las de otro. Para evitar que al cambiar el título de la *Historia de Usuario* se haya cambiado el identificador lo que haremos será dejar el ID como un conjunto de las siglas del proyecto, junto con una ***letra*** que represente la funcionalidad general de la *HU* y un ***número*** que irá en orden creciente dentro de los grupos que define la letra. Por lo tanto, para añadir un identificador a una *Historia de Usuario* habrá que mirar el número del ID más alto añadido con anterioridad a una *HU* que tuviera la misma letra y sumarle uno.

El formato de identificadores establecido quedaría como en la *Imagen 4.1*. Un ejemplo sería ***TVGB-S1*** el cual es una HU que pertenece al proyecto *TheVideoGameBox* en la que su funcionalidad principal es la búsqueda y es la primera *HU* de las búsquedas.

Imagen 4.1. Formato Indicadores

Las letras que agrupan *Historias de Usuario* por similitud en el objetivo se muestran en la *tabla 4.2*.

Función	Letra	Descripción
Search	S	Esta letra se utiliza para HU que tengan como objetivo realizar una búsqueda.
Watch	W	Esta letra se utiliza para HU que tengan como objetivo ver detalles, atributos, elementos, etc.
Login	L	Esta letra se utiliza para HU que tengan como objetivo realizar acciones relacionadas con el logueo: iniciar sesión, cerrar sesión, registro con email, registro con Google o Facebook, etc.
Order	O	Esta letra se utiliza para HU que tengan como objetivo ordenar elementos mediante algún criterio.
Game	G	Esta letra se utiliza para HU que tengan como objetivo alterar las características extra de los juegos: asignándoles un estado, aumentando sus likes, comentando una anotación-feedback en el juego, etc.
Box	B	Esta letra se utiliza para HU que tengan como objetivo alterar las boxes. Creando nuevas boxes, eliminándolas, modificándolas, añadiendo o quitando juegos, aumentando sus seguidores al seguirlas o gestionando su privacidad.

Tabla 4.2 Leyenda de las letras de HU