GREENGO

MEMORIA JPA

2018-2019

Pablo Agudo, Alejandro Cabezas, Iván Fernández,
Manuel Monforte, Gerardo Parra y Alberto Pastor

Contenido

1. INTRODUCCIÓN

1.1. CONTENIDO DE MEMORIA

Esta memoria contiene las etapas y el tratamiento que ha seguido el desarrollo del proyecto software con el nombre GREENGO. La aplicación por desarrollar recibirá el nombre de "GREENGO", la cual llevará a cabo la gestión de distintos vehículos operativos en ciudades concretas, permitiendo el alquiler de estos por clientes registrados previamente, así como la gestión de diferentes sedes, servicios contratados y empleados.

Dentro de la aplicación, gestionaremos seis módulos principales que son Vehículos, Ciudades, Clientes, Sedes, Servicios y Empleados. Dentro del módulo de vehículo diferenciaremos dos tipos, los coches y las bicicletas.

Este documento contiene los objetivos, la planificación, estructura y el modo de desarrollo del proyecto. Se tratan los cambios que se han gestionado durante la creación del producto final.

1.2. OBJETIVOS

El objetivo que presenta la realización de este proyecto es que las empresas se beneficien de nuestro software pudiendo gestionar todas las funciones especificadas en los requisitos. Nuestro usuario final es el gestor o encargado de la empresa organizadora de los eventos.

2. PLANIFICACIÓN

2.1. PLANIFICACIÓN INICIAL

Durante una primera estimación del proyecto, se planificó distribuir el trabajo entre seis personas siguiendo una estructura descentralizada democrática.

En esta segunda parte del proyecto se tomó la decisión de dividir el grupo de trabajo en dos grupos de tres personas, ambos grupos trabajando en la capa de negocio. Dentro de diseño uno de los equipos se dedicó a los diagramas de actividades y el otro equipo a los diagramas de secuencias. Dentro de codificación uno de los equipos se dedicó a los test y el otro equipo a la implementación de los diseños.

Elegimos esta estructura de trabajo por ser un grupo de trabajo pequeño, creyendo conveniente tomar las decisiones juntos y no llevar a cabo reuniones en las cuales solo participaban los jefes de los dos grupos.

En cuanto a planificación de trabajo concretamos que cada integrante trabajaría una hora diaria, descansando los sábados. Se realizarían requisitos, diseño, TDD (desarrollo guiado por *testing*) y codificación en este orden para la correcta realización del proyecto.

2.2. PLANIFICACIÓN FINAL

Durante el desarrollo del software no se hicieron reales los riesgos que indicamos en los requisitos del sistema.

En la realización del proyecto se ha seguido toda la planificación esperada menos en dos puntos que el equipo tuvo que subsanar:

- Problemas en la configuración de JPA con el entorno de desarrollo utilizado.
- Problemas con la configuración del persistence.xml a la hora de realizar testing.

Estos inconvenientes se subsanaron de las siguientes maneras:

- Para configurar JPA con el entorno de desarrollo utilizado se decidió actualizar el entorno de desarrollo a la versión intellij Ultimate, la cual si era compatible para configurar JPA.
- Se utilizó la función de la entrega anterior, para eliminar los datos de cada tabla, ya que el drop and créate daba problemas en operaciones que requerían realizar varias transaction.commit() .

3. CONCEPTOS PREVIOS

La capacidad técnica del equipo se vio limitada por la falta de conocimientos, ya que en algunas partes del proyecto eran necesarias nuevas competencias que fueron obtenidas durante el transcurso del desarrollo del proyecto.

El equipo se ha tenido que formar especialmente en los conocimientos nuevos necesarios de la asignatura de Modelado de Software. Aunque en la asignatura Ingeniería del Software se consiguieron los conocimientos básicos de las tecnologías necesarias para el proyecto, muchas de ellas se han tenido que revisar.

Estas tecnologías son: JDBC, MariaBD y su API, Java Swing, IntelliJ IDE, herramientas de SCV y el uso de herramientas CASE, en particular IBM RSA.

4. ESTRUCTURA Y DESARROLLO

El proyecto sigue una arquitectura multicapa y está estructurado en paquetes correspondientes a los módulos representados en el modelo del dominio.

En la capa de *presentación* se ha utilizado el patrón ServiceToWorker componiéndolo con el patrón Application Controller y el patrón Command. De esta forma, el Application Controller gestiona las peticiones del cliente así como las respuestas de negocio. Para tramitar estas peticiones, las empaqueta en comandos, los cuales exponen las operaciones de negocio.

En la capa de negocio, se ha utilizado el patrón Servicio de Aplicación siendo invocado por un comando mediante Factorías Abstractas implementadas como objetos Singleton. Los Servicios de Aplicación gestionan, utilizando el patrón

Transfer, las reglas de negocio apoyándose en Objetos de Negocio proporcionados por JPA, a través de anotaciones, llamados Entidades.

Para la capa de recursos se ha utilizado el almacén del dominio proporcionado por JPA conectando con una base de datos relacional implementada con ***MariaBD.***

Como herramientas de control de versiones hemos utilizado el ***Tortoise SVN*** y un repositorio de ***Git.*** Para código se ha utilizado el repositorio de Git y para los diseños utilizando la herramienta CASE se ha usado SVN.

5. PRUEBAS

Hemos realizado dos tipos de pruebas: caja negra y caja blanca. Las pruebas de caja negra se han validado desde la capa de presentación y las pruebas de caja blanca se han verificado a través del ***framework JUnit***.

Se han realizado también pruebas unitarias, es decir, de cada método para comprobar así su correcto funcionamiento.

6. DIAGRAMAS

Los diagramas realizados en el proyecto son los siguientes:

- Negocio:
  - Contract:
    - asContract(diagrama de clases)
    - asContractImp(diagramas de secuencias)
      - createContract
    - asContractFactory
  - Employee
    - asEmployee (diagrama de clases)
    - asEmployeeImp (diagramas de secuencias)
      - create_employee
      - drop_employee
      - show_employee
      - showAll_employee
      - update_employee
    - asFactoryEmployee
  - MainOffice
    - asMainOffice (diagrama de clases)
    - asMainOfficeImp (diagrama de secuencias)
      - showSalary
    - asMainOfficeFactory
  - Service
    - asService (diagrama de clases)
    - asServiceImp
    - asServiceFactory
  - Entities
    - relacionesEntidades (diagrama de clases)

- Presentación:
  - Contract
    - contractPresentation (diagramas de clases)
    - operations (diagramas de secuencias)
      - createContract
      - dropContract
      - showContract
      - updateContract
  - Employee
    - employeePresentation (diagrama de clases)
    - operations (diagramas de secuencias)
      - createEmployee
      - dropEmployee
      - showEmployee
      - updateEmployee
  - MainOffice
    - mainOfficePresentation (diagrama de clases)
    - operations (diagramas de secuencias)
      - createMainOffice
      - dropMainOffice
      - showMainOffice
      - totalSalaryMainOffice
      - updateMainOffice

  - Service
    - servicePresentation (diagrama de clases)
    - operations (diagramas de secuencias)
      - createService
      - dropService
      - showService
      - showServiceByLevel
      - updateService
  - Controller
    - Command
      - Employee (diagramas de clases)
        - commandEmployee
      - Factory
        - commandFactory
      - MainOffice (diagramas de clases)
        - commandMainOffice
      - Service (diagramas de clases)
        - commandService
      - execute
        - createClient (diagrama de secuencias)
      - generateCommand
        - generateCommand(diagrama de secuencias)
    - controller (diagrama de clases)
    - ViewDispacher
      - viewDispacherImp (diagrama de clases)

        - execute (diagrama de secuencias)
    - controller (diagrama de secuencias)

7. ENLACES

Estos son los enlaces de los repositorios de la documentación, el modelo y el código:

- Documentación: https://versiones.fdi.ucm.es/svn/MS/1819E/greengodoc
- Modelo: https://versiones.fdi.ucm.es/svn/MS/1819E/greengomod
- Código: https://versiones.fdi.ucm.es/svn/MS/1819E/greengocod

8. NOTA

Durante el desarrollo del diseño, las entidades JPA se perdieron siendo sustituidas inintencionadamente por clases UML normales.
El equipo no ha conseguido averiguar el origen del fallo, por lo que se dejó mencionado en una nota en el diagrama de clases de entidades y se siguieron utilizando las clases UML normales.

9. CONCLUSIONES

Como conclusión, el equipo ha llegado a las siguientes reflexiones:

- Los integrantes del equipo se han dado cuenta de la importancia de la realización de TDD. Con esta metodología de trabajo se localizaron antes los errores y se trabajó de manera más eficiente.
- Aunque la importancia de los requisitos y del diseño ha estado presente en todo el desarrollo, hemos comprendido cómo agiliza tener estos dos aspectos del proyectos bien entendidos y definidos.
- Como en proyectos con tiempos de entrega cortos y equipos pequeños es necesario que todos los miembros trabajen lo suficiente y aproximadamente a la misma velocidad. Este año se ha reflejado esta buena conducta en el equipo.

Como conclusión los integrantes del equipo han obtenido experiencia en todos los ámbitos del desarrollo software, en especial, en la gestión del trabajo en equipo.