Ingeniería del Software II - Enunciado del Proyecto de Curso para el curso 2022/2023

Enunciado: Pilgrimfy: Gestión de Soporte al Peregrino en el Camino de Santiago

Algunos antiguos alumnos de Ingeniería del Software 2 han hecho este verano el Camino de Santiago. Les ha gustado tanto, que han decidido dejar sus carreras profesionales y dedicarse de pleno a crear un negocio que les permita dar soporte a los peregrinos mediante la creación del servicio **Pilgrimfy**. Con Pilgrimfy, los peregrinos pueden tener acceso a varios recursos necesarios durante la realización del Camino, como pueden ser la identificación y reserva de albergues, sitios para comer, lugares donde sellar el pasaporte de peregrino, sitios donde comprar recuerdos, información de las etapas, previsión meteorológica, puntos de atención de enfermería para curar heridas…
Este servicio requiere de una herramienta **cliente/servidor**, que debería tener diferentes categorías de clientes: uno para los peregrinos usuarios, y otro para los dueños de los distintos negocios que pueden ofrecer recursos específicos a los peregrinos, tales como dueños de albergues, dueños de restaurantes, …

Para poder dar un mejor soporte, es necesario entender cómo creen los dueños de Pilgrimfy que los peregrinos usarían la herramienta. Por ejemplo, algunos prefieren reservar todos los albergues con suficiente antelación, mientras que otros prefieren ir adaptándose en el día a día e ir quedándose en aquellos que encuentren disponibles a la llegada al final de cada etapa.

Como parte de la estrategia comercial, nuestros amigos, los dueños de Pilgrimfy, quisieran tener lista para enero de 2023 una aplicación que contaran con una serie de siguientes servicios mínimos para mostrárselas a los dueños de los posibles servicios y a un grupo de posibles peregrinos experimentados que pueden validar la herramienta. Hay algunos aspectos que son fundamentales, tales como usar los servicios de geoposicionamiento para ofrecer los servicios más cercanos a la posición actual de los peregrinos, y, sobre todo, el poder explotar las capacidades de redes sociales: se espera que los peregrinos puedan valorar los negocios, proponer paradas con alto interés turístico o religioso, hacer recomendaciones de lugares donde encontrar recursos específicos, …

Los requisitos que quisieran incluir serían los siguientes para los dueños de los negocios:

- Añadir, editar, borrar, mostrar información sobre sus negocios, incluyendo los detalles más importantes.
- Leer y responder los comentarios de los peregrinos
- Hacer ofertas a los peregrinos más activos en la red (si estos han aceptado los requisitos de privacidad que les permita compartir datos específicos para poder ser localizados.

Los requisitos que se quisieran incluir para los peregrinos son:

- Planificar las etapas en función de la disponibilidad de los recursos y tiempos disponibles para el peregrino
- Mostrar información de los distintos caminos
- Buscar un tipo de recurso específico cercano a su posición actual, o bien en un punto específico del destino.
- Mostrar información sobre la etapa en curso y sobre las etapas restantes
- Ir haciendo un cuaderno de bitácora (blog) del camino recorrido permitiendo la compartición de fotos, puntos de interés, …

- Añadir comentarios a los negocios incluidos en la red

Para la propia gestión de su negocio, los propios dueños de Pilgrimfy querrían tener además cierta información, tales como:

- Número de negocios (pudiendo refinar por tipo y localización) que se han añadido al Pilgrimfy
- Número de peregrinos que han usado y que están usando la aplicación
- Número de peregrinos que han están satisfechos (al menos 4 estrellas sobre 5 en sus comentarios) con los diferentes servicios
- Análisis de las reseñas más importantes de los peregrinos sobre los negocios visitados, para poder promocionar aquellos que tengan mejores resultados….

A modo de ejemplo, se sugiere visitar algunas páginas de ejemplos reales existentes en el mercado tales como:

- https://www.pilgrim.es/
- https://santiagoways.com/es/
- ….

Trabajo a realizar

El objetivo principal del laboratorio **es la simulación del desarrollo del proyecto anteriormente descrito. Es importante tener en cuenta que deberéis elegir diferentes requisitos funcionales que podrían ser incluidos en esta primera versión[1].** Por tanto, tendréis que abordar diferentes decisiones de diseño que condicionarán el coste final del proyecto, siempre teniendo en cuenta las limitaciones que consideréis, y usando las herramientas de control, configuración y comunicación que habéis establecido para vuestro grupo de trabajo. No obstante, si por alguna razón veis que los recursos humanos y materiales que habéis establecido no son suficientes, plantead la posibilidad de incorporar nuevos recursos.

Como metodología de desarrollo se utilizará el **Proceso Unificado de Desarrollo (PUD)** que se ha explicado en clase. Haced todas las suposiciones que consideréis necesarias y adecuadas, pero explicándolas y razonándolas (todo debe quedar reflejado en la wiki del repositorio del proyecto). Las tecnologías de desarrollo dependerán de vuestra elección; en cualquier caso, la opción preferida es el desarrollo en **Java 8.**

A continuación, se indica lo que se espera de los equipos de trabajo para cada uno de los bloques, teniendo en cuenta que el ejercicio se tiene que ver como un todo.

Bloque 1

Tema 1 - Planificación y ejecución del proyecto con el Proceso Unificado de Desarrollo (PUD).

Cuando se realiza la planificación del proyecto con el PUD, los parámetros que se deben obtener son el coste final del proyecto, la agenda del proyecto y la fecha de finalización (lo que debe incluir un calendario de liberaciones o *releases*).

Por simplificar, os sugerimos hacer las siguientes suposiciones:

- Un Requisito Funcional se mapeará en un solo Caso de Uso (1:1),
- Un Caso de Uso se realizará en una y solo una iteración (1:1).
- Por simplicidad asumiremos que tendremos una *Iteración 0* en la fase de inicio, varias iteraciones correspondientes a la realización de los diferentes casos de uso, y una última *Iteración N*, que se ejecutaría en la fase de transición y donde se realizarían tareas relacionadas con el cierre del proyecto, como la integración, las pruebas de integración, el despliegue, el manual de usuario, etcétera. Supongamos que el coste de la iteración 0 es de 1000 € (sería muy interesante estimar el coste de esta iteración realmente, pero por simplicidad asumiremos ese coste), y el coste de la iteración N es de 2000 €. Tened en cuenta que esos costes son los correspondientes a recursos humanos. Sería interesante discutir si hay algún coste más que se deba incluir (en esto pueden ayudar los miembros del equipo de trabajo que estén cursando la intensificación de tecnologías de información).
- Una semana de trabajo serán 40 horas laborables.

Además, se deben observar los siguientes aspectos:

- Toda la planificación, así como todos los documentos correspondientes, actas de las reuniones, descripción de los casos de uso, documentos de análisis, ficheros *Visual Paradigm*, ficheros de imputación de horas, ...) **deben ser enlazados desde la wiki del repositorio GitHub[2]**, de modo que la wiki del proyecto sea la guía de lectura de la documentación, facilitando así la corrección[3].

[1] Se puede pensar en cada gran versión como el resultado de ciclos de PUD. Cada una de las versiones debe poder añadir un valor específico a la organización que nos ha pedido el proyecto.

[2] El Coordinador del grupo debe crear un repositorio que tenga el nombre: ISO22-ISO2-Grupo. Por ejemplo, el BC1, creará el repositorio 2022-ISO2-BC1

[3] A modo de ejemplo, os facilitaremos en Campus Virtual algunos ejemplos de años anteriores.

- Es **obligatorio que se documenten todas las reuniones que los grupos de trabajo realicen como parte del trabajo de colaboración.** Todas las decisiones / compromisos que se obtengan se mapearán a diferentes *issues* en Github, y se monitorizará adecuadamente el avance mediante el control de un tablero Kanban simplificado en un *proyecto que se creará adhoc para la práctica.* Esto servirá para marcar la trazabilidad del proyecto, y **debe haber una coherencia entre la planificación realizada, los compromisos alcanzados en la reunión, los *issues* creados, y la secuencia de avances del proyecto** (marcados por la revisión de los compromisos en los proyectos). Es decir, los pasos que deberían darse son los siguientes a modo de ejemplo:
    - En la planificación realizada en la fase de Inicio se marca un hito específico (p.ej. se creará la clase Persona.java v1.3.0 en la fecha 28/10[4]
    - En la reunión de trabajo celebrada en el 15/10 se toma y acepta la decisión que *Pepito López desarrollará la clase Persona.java v1.3.0*
    - El administrador del proyecto crea la tarea en el proyecto "Desarrollo Módulo X" en el que está la tarea *Creación de la clase Persona.java v1.3.0* y creará un *issue* "*Creación de la clase Persona.java v1.3.0"* y se asignará a *Pepito López para que haga el commit correspondiente en torno al 28/10.*
    - Pepito López trabajará a partir del día 15/10 (tras la reunión de coordinación) y hará el *commit* correspondiente en torno al 28/10 y si se diera la circunstancia el *pull request.*
    - En la sección de *Insights* de Github se podrán consultar el avance del proyecto
    - Debéis incorporar la cuenta de usuario de lo profesor de práctica que te corresponda como contributors del proyecto para que puedan monitorizar el avance del estado de las prácticas.
- Para guiar mejor el desarrollo, es altamente recomendable que se utilicen las funcionalidades de gestión de proyecto disponible en Visual Paradigm (véase Figura 3). Se recuerda que en cada iteración ***se trabajará exclusivamente en el o los casos de usos especificados[5].*** Además, y para ahorrar costes, se utilizará la funcionalidad de generación automática de código proporcionada por Visual Paradigm.
- Con respecto a la implementación, los alumnos deben "implementar" los módulos necesarios como para satisfacer todos los requisitos funcionales especificados, teniendo en cuenta la planificación realizada y el método de trabajo (iterativo, incremental, colaborativo).
- Teniendo en cuenta la aproximación 1:1 de la que partimos, se considerará que cada iteración va a generar inicialmente un componente, cuyo desarrollo debe establecerse en la planificación marcando adecuadamente las versiones que se espera ir consiguiendo – las versiones se nombrarán usando Semantic Versioning-. Al final en la última iteración se procederá a la integración de todos los componentes. Cada uno de esos **componentes se tratarán como si fuera un módulo de un proyecto Maven.** Por tanto, se deberá gestionar el desarrollo **como si fuera un proyecto multimódulo** (véase como ejemplo el capítulo 6 del libro "*Maven by examples*" (http://books.sonatype.com/mvnex-book/reference/index.html). Esta gestión puede hacerse desde la línea de comandos, o bien usando los plugins adecuados del IDE que utilicéis (recomendablemente Eclipse, con el plugin m2Eclipse).

[4] Las planificaciones deberían cumplirse en la medida de lo posible; no obstante, y como se ha dicho en las sesiones de prácticas, lo normal es que esas fechas pueda variar por diversas razones: falta de experiencia, complicaciones tecnológicas, … todas estas circunstancias son asumidas en la evaluación, pero los grupos de alumnos deben hacer un análisis sincero de las razones por las que no se ha cumplido la planificación, y proporcionar medios para paliar las desviaciones de la agenda, explicando cómo se va a resolver el problema para poder acabar a tiempo el proyecto.

[5] Es decir, que **no vale con ponerse a hacer todos los casos de uso, o hacer todos los diagramas de clases a la vez, sino cuando le corresponda según la planificación realizada.**

- El código resultante estará sujeto a la realización de las pruebas y a mantenimiento (objeto de las prácticas correspondientes a los temas 4 y 5)
- En la wiki se tiene que incluir una sección de "***Autoevaluación y Experiencia***" en donde todos los miembros del grupo incorporen una autoevaluación y una autocrítica[6] de su experiencia en el proyecto.

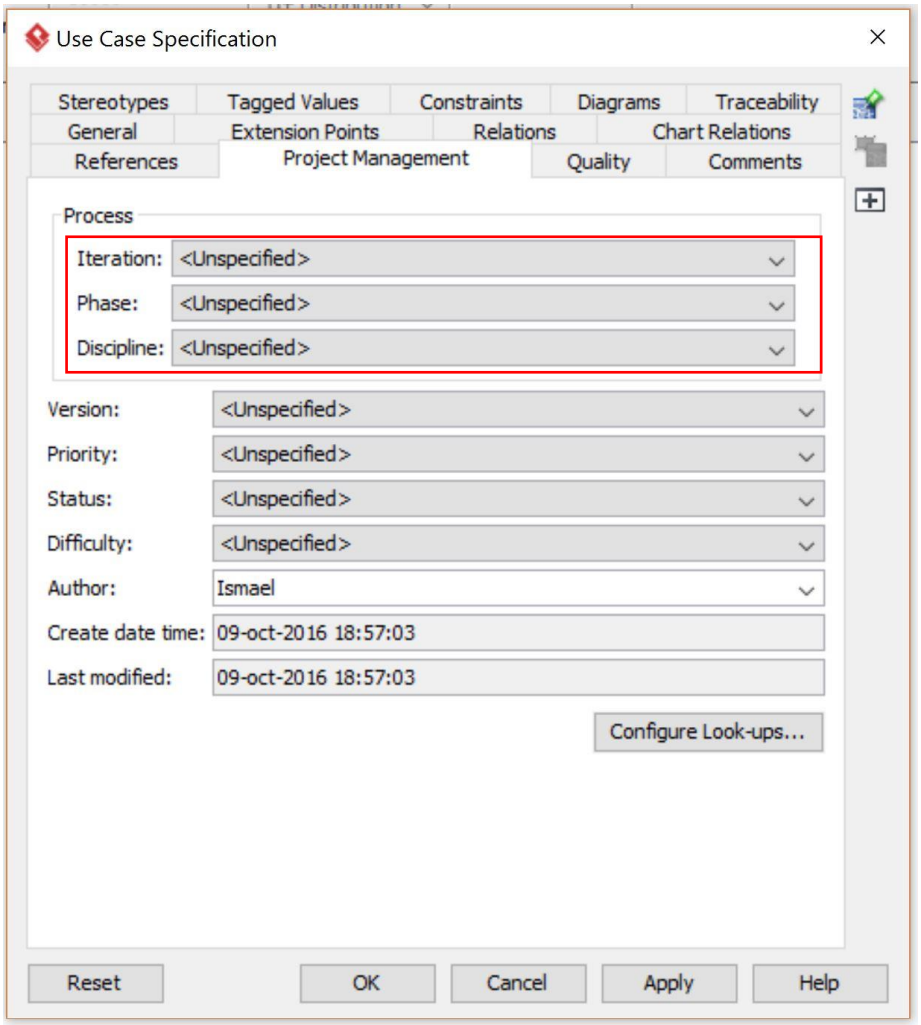

Figura 3: mecanismos de gestión de proyectos en Visual Paradigm.

Algunos aspectos importantes

La experiencia de la realización de prácticas de otros años nos ha permitido aprender una serie de lecciones que os transmitimos:

- **Aplicación del Proceso Unificado de Desarrollo**: Sería muy interesante que releyerais este enunciado una vez que haya pasado un tiempo. Uno de los principales problemas con el que nos encontramos es que los alumnos no entienden bien la asignación de casos de usos a iteraciones. Por simplicidad os recomendamos que hagáis 1RF:1CDU y 1CDU:1Iteracion. Pero esto es una recomendación -que reduce la aplicación el PUD a un ciclo en cascada - hasta que entendáis realmente que el desarrollo de un software debe agruparse en varios bloques (a los que llamamos

[6] Autocrítica no es necesariamente algo negativo: pueden ser cosas del tipo *"aprendí mucho"* o *"considero que las prácticas no me han valido para mucho"*, o *"creo que debería haber trabajado más"*, o *"ahora que hemos terminado la práctica, empiezo a entender muchas más cosas"*. En cualquier caso se trata de identificar si habéis mejorado como Ingenieros Software.

módulos) que deben cumplir la regla de máxima coherencia y mínimo acoplamiento (Principios SOLID).

- **Programación Orientada a Interfaces, no a clases**: Como consecuencia de lo anterior, uno de los principales problemas con el que nos encontramos año a año es que los alumnos no disponen de conocimientos suficientes de programación como para entender la diferencia entre programación orientada a interfaces y programación orientada a clases. La naturaleza iterativa e incremental del PUD, y el hecho de estar dirigido por casos de uso, lleva necesariamente a un planteamiento en el que el software debe ser agrupado en varios bloques (a los que llamaremos módulos), que deben ser desarrollado – de forma independiente deseablemente- en cada una de las iteraciones de la aplicación del PUD según se haya determinado en la planificación. Esta independencia obliga a un mínimo acoplamiento entre módulo que se resuelven mediante la declaración y uso de Interfaces y la aplicación del modelo multicapa (Patrón MVC – Modelo-Vista-Controlador). Cada uno de los módulos requiere paquetizar y usar adecuadamente las clases para que la comunicación se produzca a través de las interfaces correspondientes. Así que os animamos a repasar la creación de paquetes y a usarlos en el desarrollo. A fin de hacer una programación más profesional os animaríamos a usar algún framework de desarrollo tipo Spring que os simplificará mucho el trabajo una vez que hayáis superado la curva de aprendizaje mínima
- **Uso de Ramas en Github y de Maven**. Estas son de las características que más problemas suelen causar a la hora de realizar las prácticas. El problema nuevamente radica en que los alumnos abordan el desarrollo del software como un todo, sin tener en cuenta la arquitectura del sistema. Sin embargo, si los alumnos consiguieran entender que el desarrollo usando metodologías iterativas e incrementales (independientemente de si son estructuradas o ágiles) implica la descomposición de la aplicación en lo que venimos llamando módulo. Es muy importante que los módulos **no duplican el código, sino que lo usan a medida que se necesita.**
- **Abordar proyectos pequeños para entender bien los conceptos de Git y Maven.** La principal dificultad de la práctica está en asumir el cambio de paradigma (Iterativo e Incremental y Colaborativo). Si a esto se le añade la complejidad de herramientas como Git y Maven, puede llegar a ocurrir que los alumnos os sintáis confundidos y sobrepasados. Por esa razón, os animamos a que abordéis el aprendizaje de Git, Maven y otras herramientas fuera del contexto del proyecto, aplicándolo a proyectos pequeños. Una vez aprendidos y madurados estos conceptos, ya podréis aplicarlos al proyecto.

Tema 2. Gestión de Configuración

El objetivo de esta parte es la incorporación del Plan de Gestión de Configuración. La implementación de dicho plan debería quedarse reflejada en varios aspectos: quién aprueba los requisitos, quién se responsabiliza de cada componente, quién aprueba los componentes, el plan de versiones de los componentes y versiones de las construcciones.

Especialmente importante es el plan de versiones de cada componente que, debería realizarse usando la notación ***Semantic Versioning*** (http://semver.org/), y quedar reflejado de acuerdo con las recomendaciones indicadas en Git Flow, con la creación de las diferentes ramas.

Tema 3. Gestión de Calidad

El objetivo de este bloque de prácticas es la derivación de diversos requisitos tanto funcionales como no funcionales que podrían aparecer al considerar los aspectos de calidad de productos software (*quality by design)*.

Bloque 2

Tema 4. Gestión de pruebas

El objetivo para las prácticas de este tema es desarrollar y ejecutar el plan de pruebas para el proyecto. En este sentido, se deben tener en cuenta los siguientes aspectos:

- El plan de prueba debe incluir casos de pruebas para alcanzar al menos un **nivel de cobertura del 75%**
- Se generarán mediante la integración de los plugins para Surefire y Jacoco de Maven el **informe de pruebas correspondientes**, que deberá aparecer enlazados en la wiki del proyecto. Se aportará más información cuando se alcance el tema de gestión de pruebas.

Tema 5. Gestión de mantenimiento

Tomando como punto de partida el informe de pruebas obtenido en el informe anterior se trata de corregir los errores en el código y o de arreglar algunos aspectos de limpieza del código (*clean code)* Para ello se realizará un plan de mantenimiento enfocado al mantenimiento del sistema, que partirá de la creación de un informe automatizado generado con Maven mediante la inclusión de los diferentes plugins: **PMD/Findbugs**. Se aportará más información cuando comiencen las prácticas del Tema 5.

Herramientas recomendadas necesarias.

1) Entorno Eclipse con plugins de Maven y de Git.
2) Servidor de base de datos MySQL (o máquinas virtuales con los SGBDRs desplegados y los puertos correspondiente del firewall abiertos).
3) Cliente de base de datos MySQL Workbench
4) Visual Paradigm

Hay abierta una wiki en la sección del proyecto para que podáis enlazar el enlace al repositorio[7] github de vuestro proyecto.

[7] Recordad que Github no permite borrar nada, por lo que todo lo que subáis, se quedará en vuestro repositorio para siempre… en su lugar y para hacer pruebecillas, os recomiendo que uséis otros repositorios auxiliares.