Proyecto de Desarrollo del Software
Práctica 2 - Sprint 0

Presentación

Una vez conceptualizado y planificado el producto y con los requisitos en forma de épicas estimados y priorizados, dentro de la Actividad 3, pasamos al desarrollo propiamente dicho del primer sprint (cero) de inicialización, la configuración de herramientas y la creación de un primer esqueleto del producto, fruto del diseño, implementación y despliegue de un prototipo inicial.

Para elaborar esta PRAC2 hay que repasar los materiales de las Actividades 1 y 2 (ver Plan Docente y Recursos del aula) y consultar los recursos del Laboratorio para el software necesario para el desarrollo de esta práctica. Continuaremos el trabajo en grupo.

Competencias

En esta Práctica se trabajan las siguientes competencias del Grado en Ingeniería Informática:

- Trabajo en equipo.
- Capacidad de diseñar y construir aplicaciones informáticas mediante técnicas de desarrollo.
- Aplicación de las técnicas específicas de la Ingeniería del Software a las diferentes etapas del ciclo de vida de un proyecto.
- Capacidad para proponer y evaluar diferentes alternativas tecnológicas para resolver un problema concreto.

Objetivos

Los objetivos concretos de esta Práctica 2 son:

- Gestionar e intercambiar la información en el equipo en un entorno virtual.
- Facilitar la comunicación y la interacción entre los miembros del equipo virtual.
- Optimizar el tiempo en un entorno de trabajo en equipo virtual.
- Estimular la toma de decisiones y gestionar los conflictos en un trabajo de equipo virtual.
- Identificar los modelos y métodos más importantes y estándares para el desarrollo ágil de software.
- Exponer argumentos sobre los puntos fuertes y los puntos débiles de las diferentes tecnologías implicadas en el desarrollo y despliegue y escoger las más adecuadas para el producto.
- Implementar microservicios centrados en la interfaz de usuario, la lógica de negocio y el acceso a datos.
- Reconocer los factores de entrega, despliegue e integración continúa.
- Aprender las actividades básicas en el desarrollo de un producto de software.

Descripción de la práctica a realizar

Los responsables del producto **Photo&Film4You** han evaluado la planificación y el análisis de los requisitos de alto nivel (épicas) del producto, y de nuevo han quedado muy satisfechos. A partir de vuestra planificación en la PRAC1, en esta PRAC2 realizaremos el ***walking skeleton*** del producto (en algunos contextos esta actividad está incluida en lo que se denomina *Sprint 0*).

Para abordar la práctica, hay que configurar los equipos y tareas para llevar a cabo las actividades previstas (nota: no es necesario un PO en esta práctica). Revisad las referencias del apartado Recursos y usad las mismas herramientas de la PRAC1, más otras herramientas nuevas como draw.io, GitLab, IntelliJ, SonarQube, Docker, así como los materiales de ISCSD (consultad el Laboratorio).

El concepto ***walking skeleton*** fue descrito por primera vez por Alistair Cockburn (2004), uno de los fundadores del desarrollo ágil, como: *"...is a tiny implementation of the system that performs a small end-to-end function. It need not use the final architecture, but it should link together the main architectural components. The architecture and the functionality can then evolve in parallel.".* Podéis encontrar más información en el artículo: "*Walking Skeleton: beginners tips*" de Daniele Scillia (ver Recursos).

Siguiendo estas referencias, en esta práctica implementaremos el *walking skeleton* del sistema **Photo&Film4You** que solo tendrá una prestación pequeña. No tiene porque usar la arquitectura final, pero tiene que tener una primera conexión entre todos los componentes de la arquitectura del sistema, para que la arquitectura y la funcionalidad puedan crecer y evolucionar de forma conjunta en los siguientes sprints. Para lograr este propósito, os proponemos implementar la funcionalidad "**Crear Alerta**" a partir del material de ISCSD.

El ***walking skeleton*** que os pedimos tiene que incluir:

1. [10 % de la nota] Definición del modelo de versionado y branching. Justificad el modelo escogido.

   NOTA: Para abordar este ejercicio recomendamos la lectura del artículo "*A successful Git branching model*" de Vincent Driessen y el artículo complementario "*What Are the Best Git Branching Strategies*" de Rowan Haddad (ver Recursos).

2. [20% de la nota] Diseño de la arquitectura base para la funcionalidad Crear Alerta:

   - Representad visualmente en uno o más diagramas los diferentes elementos implicados en la funcionalidad (tanto la parte de la aplicación Web como de los microservicios y su integración).

   - Justificad en el documento entregado las decisiones tomadas y los patrones escogidos (factories, adaptadores, command handlers, etc.). Se recomienda el uso de *domain-driven design* (repasad DDD en el Cap. 5 "*Designing business logic in a microservice architecture*" del libro "*Microservices Patterns*" de Chris Richardson de los materiales de ISCSD que podéis encontrar en Recursos).

- Identificad y justificad a alto nivel las tecnologías o frameworks escogidos para desarrollar cada parte de la arquitectura (frameworks para el frontend, librerías de backend, etc.).

3. [50%] Implementación y testing de la arquitectura base aplicada a la funcionalidad Crear Alerta que servirá de referencia para construir el resto de funcionalidades. La implementación debe incluir todos los componentes y elementos descritos en el diseño de la arquitectura del ejercicio anterior:

    - Implementación del frontend. Debe incluir los campos necesarios para recoger la información relativa a una alerta. Implementad validaciones básicas para asegurar que se cumplen todos los campos necesarios.

    - Implementación del microservicio.

    - Tanto en un caso como en el otro, debéis programar y entregar también los **tests automáticos** unitarios y de integración asociados a la funcionalidad de Crear Alerta.

    - Para poder validar la implementación de la funcionalidad, tendréis que entregar un entorno autocontenido, creado con ***docker-compose***. Este entorno debe contener el arranque de todos los servicios de plataforma necesarios para correr la aplicación y los servicios.

    - Proporcionad un conjunto claro de instrucciones que detallen cómo arrancar y detener la aplicación usando *docker-compose*, así como cualquier otro paso necesario para configurar o inicializar el entorno (Consultad el Laboratorio para más información). Es **requisito imprescindible** que la aplicación arranque correctamente y sea accesible siguiendo vuestras instrucciones. Adjuntad capturas de pantalla que muestren también los diferentes pasos y una ejecución de la funcionalidad.

    NOTA: Se trata de un *walking skeleton*, por lo que la arquitectura no tiene porque ser la definitiva si hay algunos aspectos que aún se deben refinar o validar. Para abordar este ejercicio, tomad como base el diseño y la implementación de “Photo&Film4You” de ISCSD (consultad el Laboratorio).

4. [20% de la nota] Definid e implementad una versión inicial del pipeline que asegure para la funcionalidad desarrollada:

    - Obtención del código de las ramas de desarrollo (GitLab).

    - Compilación.

    - Análisis de la calidad del código (con Sonarqube). Incluir una valoración del resultado en el informe a entregar.

    - Ejecución de las pruebas automáticas y medida de la cobertura. Para abordar este ejercicio, revisad el capítulo 5 “*Anatomy of the Deployment Pipeline: Implementing a Deployment Pipeline*” del libro de Farley and Humble (ver Recursos).

Qué hay que entregar

Al final de la **Práctica 2** cada grupo tendrá que entregar en su espacio de grupo del Drive, dentro de la carpeta *Documentos para el Consultor,* un solo documento PDF con las respuestas a los 4 ejercicios pedidos en un máximo de 25 páginas (de forma orientativa, se propone una distribución de páginas por cada ejercicio):

1. Informe justificativo del modelo de versionado y branching elegido (máx. 3 páginas).
2. Diagramas de la arquitectura propuesta y justificaciones (máx. 5 páginas).
3. Implementación (máx. 5 páginas):
    a. Referencias a los repositorios de código del desarrollo y archivos necesarios para desplegar la solución con docker-compose.
    b. Pasos para la instalación de la aplicación con docker-compose y capturas de pantalla.
    c. c. Capturas demostrativas del funcionamiento de Crear Alerta.
4. Pipeline (máx. 12 páginas):
    a. Referencia al pipeline (enlace). Asegurad previamente que los consultores tienen permiso de acceso.
    b. Incluir capturas demostrativas de su funcionamiento (máx. 10 páginas).
    c. Incluir un análisis de los resultados obtenidos con SonarQube.

NOTA: Para facilitar la realización de esta práctica, junto con el enunciado, os proporcionamos una solución de la misma práctica de un curso anterior. Esta solución se basa en un caso de estudio diferente y contiene algunos ejercicios diferentes. Es importante que la consideréis como una guía orientativa y no como un patrón exacto a seguir. Su objetivo es ayudaros a comprender mejor las expectativas y orientaros en la estructura y enfoque que puede tener vuestra solución.

Recursos

Recursos Básicos

- Artículo "Walking Skeleton: beginners tips" by Daniele Scillia. https://medium.com/dan-the-dev/walking-skeleton-beginners-tips-deef5baebb5b
- Artículo “A successful Git branching model” by Vincent Driessen. https://nvie.com/posts/a-successful-git-branching-model/
- Artículo “What Are the Best Git Branching Strategies” by Rowan Haddad. https://www.abtasty.com/blog/git-branching-strategies/ (complementa el artículo de Vincent Driessen).
- Libro “Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation”, caítulo. 5: “Anatomy of the Deployment Pipeline: Implementing a Deployment Pipeline” by David Farley and Jez Humble. https://learning.oreilly.com/library/view/continuous-delivery-reliable/9780321670250/ch05.xhtml
- Libro “Microservices Patterns”, capítulo 5: “Designing business logic in a microservice architecture” by Chris Richardson (material d’EPCSD): https://learning.oreilly.com/library/view/microservices-patterns/9781617294549/OEBPS/Text/05.html#ch05lev2sec3

Recursos Complementarios

- Libro "The DevOps Handbook", Part I: “The Three Ways” Capítols 1-4 by Patrick Debois, John Willis, and Jez Humble, Gene Kim. https://learning.oreilly.com/library/view/the-devops-handbook/9781457191381/DOHB-pt_01.xhtml
- Libro “Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation”, Capítulo 14: “Advanced Version Control” by David Farley and Jez Humble. https://learning.oreilly.com/library/view/continuous-delivery-reliable/9780321670250/ch14.xhtml

Criterios de evaluación

- El documento-solución de la **Práctica 2** será común para todos los miembros del grupo, pero se evaluará individualmente en base a la aportación individual.
- El coordinador tiene que hacer un ejercicio realista de la valoración del trabajo en grupo con un **informe privado** a entregar en el apartado "Entrega de la PRAC2" del aula PDS del Campus de la UOC (al que solo tiene acceso el Consultor) y un **informe público** accesible por el grupo a entregar en el espacio del grupo del Drive, dentro de la carpeta *Documentos para el Consultor*.
- Ambos informes tendrán que estar correctamente justificados, siendo la calidad y precisión de esta justificación un aspecto fundamental en la evaluación del coordinador.
- Hay que tener en cuenta que durante todas las pruebas del curso el coordinador de cada prueba tendrá que publicar las actas de las reuniones virtuales.

Formato y fecha de entrega

- Un único documento PDF con la solución de los ejercicios pedidos (no hay que entregar por separado ficheros con gráficos ni similares).
- Los **dos informes** que tiene que redactar y entregar el/la Coordinador/a según se ha indicado más arriba (encontraréis las plantillas en el Tablón de PDS del Drive).
- Todos los documentos se tienen que librar en los espacios indicados más arriba antes de las **23:59 horas del día 28 de noviembre de 2023**. No se aceptarán entregas fuera de plazo.
