Proyecto

de CFGS

2º DAM

Tabla de contenidos

Presentación

El presente anteproyecto pretende ser un nexo de unión entre el material impartido a lo largo de estos estudios y la situación actual que nos ocupa, tanto a nivel social como tecnológico. Es decir, se trataría de un desarrollo de una aplicación web *full stack* modularizada lo más completa y funcional posible; ya que esto permitiría trabajar, implícitamene, la mayoría de las habilidades requeridas en el proceso de desarrollo del software.

Por ende, el título de este trabajo podría identificarse como “flexibilización de los aplicativos en el marco laboral”.

Objetivos y forma

Uno de los mayores problemas que se ha tenido durante años es que a la complejidad intrínseca del software, se une la rapidez con la que evoluciona la tecnología y la cantidad de posibilidades existentes. Sin embargo, esto tiene otro lado positivo que deriva en mejoras en la eficiencia de los equipos de hardware y en la manera en que se entiende el software. Tal es el caso de los “contenedores” y la programación orientada a servicios.

Con respecto a lo primero, es cierto que ese concepto de virtualización había sido conocido en el mundo de Linux; no obstante, se revolucionó con la llegada de Docker en el mercado. Tanto es así, que incluso se ha estandarizado su uso y cada vez es más común encontrarlo tanto a nivel de desarrollo como en producción; pues mejora la compatibilidad entre equipos (al tratarse de una

capa de software lo hace más predecible) y abarata muchos costes (facilita la escalabilidad según se necesite y es más ligero una virtualización convencional).

En cuanto a los servicios, aprendidas las lecciones de la época de la profunda crisis del software en los años 80, las arquitecturas monolíticas han ido perdiendo fuelle para dar paso a otro tipo menos solapada, la de orientada a los servicios. Principalmente se ha venido trabajando con SOA; pero en determinados proyectos es necesario una granularidad más fina, los llamados microservicios. Aunque puede haber cierto debate acerca de los límites entre estos dos o los de los microservicios con los nanoservicios, la idea que pretendo trabajar considera la necesidad de separar funcionalidades de una manera simple para ajustarse a los cambios de la tecnología y los procesos de negocio. Por este motivo, puedo concretar que me enfoco más a conseguir una arquitectura más orientada a servicios que la tradicional monolítica.

Alcance del proyecto

Uno de los mayores problemas que se tiene en el desarrollo de software es el entendimiento con el cliente, pues a veces existe un gran lapso entre el conocimiento de ambos y de sus objetivos. Debido a ello, pretendo abstraer toda la lógica de la “negociación” con el cliente, para así ser capaz de adaptarme a lo que vaya necesitando. Algo así como una filosofía más tipo Kanban.

Por tanto, de cara a lo que verá el usuario final, este proyecto pretende ofrecer:

- Una página web estática de la empresa.

- Acceso a un área de usuario donde podrá introducir los pedidos de los clientes y persistirlos (para enviarlos al almacén por ejemplo).
- Una opción que permitirá ver los datos guardados en la BBDD.

¿Por qué estas funcionalidades?

El mundo está más tecnificado y, por ciertas ventajas competitivas, más centrado en la nube. Como consecuencia, se pretende ofrecer una solución que permita acercar al usuario a su uso, a la vez que se pretende dar flexibilidad de plataforma para que pueda realizar sus tareas rutinarias laborales en un ambiente confortable. Podrá hacerlo desde una tablet, móvil, pc,…
Además, se evitará el extravío de documentos, facilita la digitalización de la empresa, prevendrá malentendidos en el equipo y ahorrará otro tipo de costes. Cosas que se consideran que aportan valor al producto final.
De este modo, al digitalizarse se puede hacer un seguimiento de las ventas con mayor precisión lo que ayudará a tomar mejores decisiones.

Propuesta de trabajo, metodología y planificación temporal

Teniendo en cuenta que el plazo temporal será de dos meses aproximadamente, podría dividir el trabajo en bloques de una semana, de este modo:

- Semana 5 abril → continuar con la planificación del proyecto y hacer un estudio básico de las tecnologías.

- Semana del 12 → montar el backend.
- Semana del 19 → BBDD, margen para ajustar el backend o adelantar el frontend.
- Semana del 26 → frontend.

---------- Algo entregable – feedback cliente ----------

- Semana del 3 mayo → ajustes, añadir funcionalidades backend-fronend.
- Semana del 11 hasta cuando sea el final → “ ”

Nota: a partir de que esté montada la funcionalidad básica, se pretende trabajar más hacia un modelo de CI/CD. Esto es, se tenderá a pensar en *features*, ampliar funcionalidades y mejorar existentes. Por ejemplo, se puede mejorar la seguridad, crear gráficos, crear un canal para mensajes,...

A simple vista, se puede observar que se trata de un flujo de trabajo en el marco de las metodologías ágiles, al principio más iterativo que continuo y a medio camino entre el estilo know-how y de procesos (por falta de conocimientos técnicos por mi parte, me apoyaré en la tecnología). Así que, para la primera etapa será una especie de Scrumban y XP .

Una vez que se ha aclarado el esquema temporal, parecería sensato hablar de las tecnologías que finalmente pretendo usar:
El backend va a estar desarrollado en Java y usaré Spring/Springboot, la BBDD será PostgreSQL, el frontend JS junto con React/Redux y tal vez Bootstrap, HTML y CSS.

Para el IDE probaré con Visual Studio Code, como herramienta de gestión usaré Maven, control de versiones Git/Github y también trataré de usar Docker,

¿Por qué esas elecciones? Punto de vista personal

Java ha sido el lenguaje principal que se ha estudiado y me parece muy multifuncional, por lo que un estudio más profundo de él puede ser beneficioso, aprender de sus estructuras, expresiones,… Spring parece un software impresionante lleno de módulos, en concreto Springboot facilita el uso Restful porque además integra el servidor; pero también habría que destacar el ORM Hibernate. Me llaman la atención otros como Webflux o Spring Security. En cualquier caso, con los primeros mencionados ya creo que tendré bastante material de estudio para aprender.

La base de datos PostgreSQL me ha llamado mucho la atención desde el primer momento que empecé a trabajar con ella, la manera en que se organiza parece ser fascinante. Además está muy soportada por desarrolladores, lo que puede dar pie a entender funcionalidades, a veces, más complejas.

JavaScript, va a ser un reto trabajar con él y sus frameworks, pero estoy seguro de que su estilo más funcional me ayudará a crecer como desarrollador. React parece una buena opción de cara a la división en componentes, lo que puede ser útil para independizar funciones. La librería Redux puede ser un soporte interesante de cara a otros proyectos más avanzados. Bootstrap, si tengo

tiempo de visualizarlo, ayudará a mejorar la adaptación multiplataforma (si no tendría que hacerlo más rudimentario o enfocarme en un tipo concreto de inicio).

HTML y CSS será simplemente por la estructura básica y dar un poco de estilo que facilite la usabilidad de la aplicación.

La proposición de cambio del IDE es que se me ha estado quedando colgado Eclipse, NetBeans nunca me ha funcionado muy bien e Intellij IDEA es similar a Android Studio; así que creo que me consumirá muchos recursos. También puede ser útil familiarizarme con otros entornos de trabajo.

Maven, aunque no creo que llegue a profundizar tanto como para hacer composiciones complejas del POM, el usarlo desde línea de comandos puede hacer que entienda mejor los ciclos en el desarrollo de software a través de sus *goals.*

De Git es sabida su importancia en el mundo de la programación, incluso por temas conceptuales puede enriquecer mi visión de la integración de las cosas. Github, como extensión del concepto anterior, como vista a trabajo distribuido.

Docker, creo que ayudará a entender la independencia entre procesos. Al instanciar las imágenes se consumen menos recursos que con las máquinas virtuales.

Presupuesto

Tomando como punto de partida la cantidad tan grande de tecnologías que se pretenden utilizar y la metodología a seguir, me parece justo calibrar 20€/hora de servicio trabajado. Sumando el 10% recomendado se quedaría en 22€/hora de base.

En cuanto a equipo técnico no se requiere gran cosa en un primer momento, pues dependería de lo que necesitara el cliente. Como es un proyecto ficticio, supondré un cliente meta que esté barajando la posibilidad de digitalizarse y quiera investigar con nuevas tecnologías pero no tenga mucha carga de trabajo. Por lo que se usarán los recursos de los que disponga y se virtualizará con Docker esos espacios. Así, se consigue una solución independientemente de los SOs que tenga instalado a la vez que me puedo cerciorar de que si funciona en mi entorno puede funcionar en el suyo también.

El siguiente tipo de cliente sería uno que se quiera integrar con un proveedor de servicios en la nube como Azure, Google Cloud, Oracle o AWS. Pero no me parecería lo más apropiado pensar en esos tipos de deployment de momento, porque si necesita de ese servicio es porque tiene grandes volúmenes de trabajo, y si los tiene, no me veo capacitado aún para trabajar a nivel de seguridad con esos datos.

Fuentes principales de documentación a usar

- Tutorial de Oracle para Java.
- Página web Spring.io
- Página Hibernate.
- Documentación oficial Docker.
- Documentación oficial Git.
- Web de Freecodecamp
- Web de theodinproject
- Javatpoint
- Píldoras informáticas (canal youtube)
- JavaBrains (canal youtube)
- Documentación de RedHat, Microsoft, IBM, …
- W3Schools
- Developer.mozilla
- Curso Web development with Java Spring franework – en web coursera
- Curso Continous Delivery & DevOps – en web coursera
- Web springboottutorial
- CCNA v7 de Cisco
- Documentación PostgreSQL
- Aplicación sololearn

Síntesis

A modo de resumen, especifico que se trata de un proyecto web de corte empresarial. Se podría pensar como un pequeño módulo de un ERP pero orientado a ser funcional en la nube (del tipo que sea).