1. Postevaluaciones y conclusiones

1.1. Postevaluaciones

1.1.1. Postevaluación de la arquitectura

La arquitectura inicialmente planteada trataba de un diseño cliente-servidor desacoplado, con ambos componentes siendo monolíticos de microkernel con un núcleo particionado, debido a que los servicios queríamos que fueran despoblados y fácil de añadir o quitar. Este objetivo se ha conseguido mantener hasta cierto punto, dependiendo de qué sección del proyecto se quiera observar.

Dentro del apartado del servidor **Java** esto si se ha cumplido completamente, hasta el punto que incluso se pueden añadir y quitar servicios sin tener que recompilar el núcleo, basta con re ejecutarlo para que encuentre los nuevos servicios. Esto es más que probable que se deba a nuestra experiencia previa con **Java**, la cual ya llevamos años desarrollando y no es ningún cambio brusco de paradigma. También pensamos que ha sido desde luego una decisión perfectamente alineado para el tipo de proyecto que se debía desarrollar, ya que uno puede describir nuestra arquitectura del mismo modo que la aplicación. Un hub o núcleo con múltiples servicios o plugins descapotables.

Por desgracia, aunque como uno podría suponer, el cliente estando basado en **JS** y **React** no ha podido seguir tan estrictamente esta idea. Tratándose de un lenguaje totalmente nuevo y un *framework* también nunca visto para nosotros que da por sabido el lenguaje en el que está basado, ha sido todo un desafío por sí mismo. Igualmente, pensamos que no debería costar mucho adaptarlo para que esta parte también siguiera la arquitectura propuesta, sin embargo esto es algo que no hemos podido abarcar por completo dentro del tiempo permitido.

De todos modos, esto no es ningún problema ya que se planificó desde un principio que esto pudiera ocurrir. Por eso siempre se planteó la parte cliente como ultra-ligero y sin lógica de negocio alguna, siendo posible evaluar el servidor de manera aislada si se quisiera, cumpliendo todos los requisitos pedidos. Es por esto mismo que este publica una *API* con todas las acciones y vistas necesarias; como añadir o quitar servicios o ubicaciones, y ver noticias, eventos o el clima de estos. Simplemente con una interfaz algo más orientada a otros desarrolladores en vez de usuarios finales. Algo que también abre las puertas a poder reescribir y rediseñar el cliente fácilmente usando otras tecnologías o lenguajes.

1.1.2. Postevaluación de la interfaz

Consideramos que la interfaz conseguida es plenamente funcional y cumple con todo lo que se le puede a exigir a una aplicación con estas características.

Como punto clave de nuestro diseño de interfaz nos gustaría resaltar la usabilidad sin registro. Desde un inicio, por la naturaleza de la aplicación. La percibimos como una web de consulta rápida y creímos que era necesario, primero ofrecer a los usuarios una versión de invitado que permitiese consultar la información directamente, evitando así tener que pasar por ningún tedioso sistema de registro y que el usuario ya posteriormente, cuando él lo desee,

pueda crear una cuenta para conseguir persistencia en sus ubicaciones. La guinda a todo este proceso de sistema de invitados, es que cuando un invitado crea una cuenta, todos los lugares guardados en su sesión de invitado son directamente volcados a la nueva cuenta creada. De modo que no pierde los lugares que previamente había guardado para estar al corriente de sus servicios.

Otros puntos que queremos destacar es la responsividad del diseño logrado, adaptándose al tamaño de pantalla de cualquier dispositivo y siendo una web app lista para usar tanto en un navegador de ordenador como en un dispositivo móvil smartphone o tablet.

Finalmente mencionar que creemos que el diseño es armonioso, agradable a la vista, fácil de usar e intuitivo, siendo todos estos puntos, las claves sobre las que se ha sustentado el desarrollo y los que hemos perseguido en todo momento.

1.2. Conclusiones

1.2.1. Cumplimiento de objetivos

Consideramos que el proyecto ha cumplido y excedido todos los requisitos básicos pedidos para su desarrollo, por lo tanto este proyecto debería ser clasificado como avanzado, teniendo la posibilidad de optar a la nota máxima. Presentamos a continuación los objetivos adicionales propuestos a ser considerar.

La primera de ellas es que es una aplicación multiusuario y multiplataforma, ambas añadiendo complejidad y funcionalidad que inicialmente no estaban contempladas. Estas hacen necesario la introducción de protocolos de seguridad al gestionar los datos personales como las contraseñas, modelos relacionales más complicados y la necesidad de que la aplicación sea responsiva y adaptable a cualquier tipo y tamaño de dispositivos.

La segunda es el aspecto cuentas de invitado y el autocompletado de búsquedas, ambas de nuevo añadiendo otro grado de dificultad sobre esta. Estás hacen necesario que la arquitectura interna permite tener múltiples estados, ya que añade diferentes tipos de usuario; además de que se haga un uso eficiente de recursos, ya que autocompletar cualquier tipo de ciudad y coordinada no es un proceso sencillo.

1.2.2. Experiencia personal

El proyecto, no sin incidentes, ha sido concluido dentro del plazo estipulado y ha alcanzado las expectativas de calidad de los miembros del equipo, por lo que se podría decir que el proceso ha sido exitoso, a continuación detallaremos en profundidad las reflexiones de los miembros del equipo respecto a los punto más relevantes del trabajo realizado durante estos meses de desarrollo.

El primer punto a detallar serán las experiencias y reflexiones sobre el que es el tema principal de una de las asignaturas que forman este proyecto, la dinámica *ATDD*.

Por consenso, los miembros del equipo podemos afirmar que hemos encontrado este estilo de desarrollo interesante y con mucho potencial. Aunque cabe mencionar que también lo hemos encontrado, a veces, limitante a la hora de trabajar con él. Debido a que ofrece muy poca flexibilidad frente a los cambios de diseño que surgen durante el desarrollo. Entendemos que esto que nosotros hemos detectado como un inconveniente es una de las principales ventajas de este tipo de desarrollo, pero creemos que un proyecto llevado a cabo por gente con poca experiencia no es el grupo de trabajo que más se puede beneficiar del desarrollo *ATDD*. Ya que por la naturaleza del grupo es habitual encontrarse en situaciones donde la idea previa sobre la que se estaba basando el trabajo es o bien incorrecta o subóptima.

El segundo punto sería respecto a las motivaciones y metas adicionales que nos hemos propuesto en el desarrollo del proyecto.

Uno de estos era la gestión o organización de proyectos, ya que en experiencias previas personales y académicas el principio siempre era algo desorganizado, trayendo problemas en las fases posteriores del desarrollo. Entonces por este motivo decidimos primero indagar y aprender más sobre **Maven** y **Git**, ya que son herramientas claves para solucionar esto. Utilizarlas y saber cómo funcionan ha terminado siendo una muy buena decisión, ya que al contrario que otros proyectos conseguimos optimizar la experiencia posterior de desarrollo. Por ejemplo, descubrimos cómo desacoplar diferentes capas lógicas de la aplicación en repositorios diferentes e independientes; también conseguimos que el preparar el entorno de despliegue para ejecutar el proyecto no sea más que ejecutar un solo comando o un simple clic en un editor, algo que en cualquier proyecto antes realizado podía llegar a ser horas de frustración intentando averiguar cómo obtener todo lo que te pudiera hacer falta.

Otra clave en la que el equipo ha puesto gran empeño ha sido el aprendizaje de una nueva herramienta cómo ha sido **React**. En general las valoraciones del grupo hacia esta herramienta son extremadamente positivas y todos los miembros estamos muy satisfechos con haber elegido esta herramienta para llevar a cabo el desarrollo de la vista. La elección del *framework* de *frontend* a utilizar no fue una tarea fácil, pero creemos que tomamos la decisión correcta y si hoy tuviésemos que volver a elegir un *framework* de js volveríamos a elegir **React**. Como puntos clave el equipo quiere resaltar las grandes comodidades que ofrece a la hora de renderizar o actualizar componentes y gestionar el flujo de información frente al desarrollo en **JS** puro. Finalmente podemos decir, que en conjunto hemos conseguido aprender lo básico de **React**, construyendo una buena base que nos permitirá, en proyectos futuros, poder utilizar esta herramienta con solvencia. Como anécdota a este punto cabría mencionar que en un primer momento la perte de **React** la planteamos para desarrollarla utilizando algunas de las más grandes y populares bibliotecas, como son **React-Router** y **React-Redux**, pero por motivos de tiempo y alcance decidimos eliminar estas bibliotecas del proyecto y centrarnos en dominar lo básico que la herramienta ofrece que ya es mucho.

Finalmente, como lenguaje vehicular para compartir el conocimiento que hemos ido desarrollando a lo largo del proyecto y poder hacer un documento de memoria a la altura, hemos elegido LaTeX, que es el lenguaje con el que se ha constituido la totalidad de esta documentación. Al fin y al cabo, una memoria como es el caso, se realiza pensando en todo momento como transmitir el conocimiento a una tercera persona (que no tiene porqué ser

especialista en la materia), y ese es el motivo y el anhelo de la extensión y reflexiones en la mayoría de los pasos que se han ido dando a lo largo de todo este documento. El poco a poco familiarizarse con este lenguaje, no ha sido una tarea para nada sencilla, puesto que la curva de aprendizaje es bastante elevada, pero sin lugar a dudas merece la pena.

LaTeX entre otras cosas, permite citar de forma casi automática fuentes, con el control de bibliografías, pies de página, pies de figuras, etc. Este lenguaje es óptimo para realizar tareas como esta, no obstante, en documentos largos como es el caso, plantea algunos problemas. Aunque el documento, internamente esté fraccionado en 7 subarchivos `.tex` y su compilación por secciones pueda ser local a cada archivo para ver cambios, cuando se desea tener el documento general, o bien que estos subarchivos hereden paquetes o configuraciones nuevas, es preciso compilar el documento entero y esto plantea una pérdida de tiempo interesante. Lo es, porque el motor `pdftex` que compila y genera el `pdf` no está programado para que use varios núcleos a la vez, y por lo tanto, todo el trabajo recae sobre un mismo núcleo. El tiempo estimado de compilación del documento entero (con casi 300.000 carácteres) es de casi 1 minuto, ejecutando esta compilación en un procesador bastante reciente: Apple M1 Pro. Cuando se requiere realizar un cambio y que su impacto sea global en todo el documento (cambio de `packages`, por ejemplo), hay que esperar cada vez que se compila un cambio a ver el resultado, y a la larga, ralentiza el proceso de documentación.

Para finalizar, el poder trabajar con compañeros tan comprometidos en todos los aspectos en este proyecto, facilita muchísimo el trabajo y el avance en el mismo. Hemos aprendido formas nuevas de desarrollar, *frameworks*, etc. No ha sido un camino fácil, pero sí muy gratificante.

Ahora sí, `\end{document}`