# Registro de Decisiones

Se considera importante registrar las decisiones para hacer un seguimiento de la evolución de la solución, facilitar su compresión y justificar las decisiones.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fecha | Decisión | Contexto | Ventajas | Desventajas | Alternativas |
|  | Hicimos un Strategy con los puntos de interés. | Modelo Estático  Diagrama de Clases | Nos permite definir el comportamiento de cada tipo de POI | Mayor cantidad de clases |  |
|  | Se añadió una superclase que engloba a tanto los pois como a los servicios. | Modelo Estático  Diagrama de Clases | Nos permite que tanto pois como servicios compartan la generación de id y que en la base de datos los horarios puedan hacer referencia a una sola tabla ya que componen a muchos clases distintas (pois, servicios de cgp, servicios bancarios) |  | Implementar distintos tipos de horarios para pois y servicios |
|  | Los horarios se diferencian en horarios regulares (día de la semana y rango horario) y en horarios especiales (fecha y rango horario). | Modelo Estático  Diagrama de Clases | Nos permite contemplar una distribución de días de atención tanto semanal como para feriados, días festivos, etc | Mayor complejidad ya que se necesitan dos clases distintas con distinto comportamiento. |  |
|  | Interfaz DateTimeProvider para obtener fecha | Modelo Estático  Diagrama de Clases | Inyección de dependencias, para no hacer “new Date()” dentro del método. Esto nos permite hacer el método testeable. | Usar un constructor custom | Hacer el new dentro del método y mockear la generación de la fecha |
|  | Singleton en terminal interactiva. Pensamos en sistema por máquina | Modelo Estático  Diagrama de Clases |  | Conflicto con entrega 3 y búsquedas | Considerar la terminal como un tipo de usuario |
|  | Archivo properties para especificar parámetros configurables | Modelo Estático  Recursos | Permite modificar un sólo lugar | no es fácilmente modificable desde la aplicación | Guardarlo en la base de datos |
|  | Búsquedas almacenadas en archivos con formato JSON por no contar con BD. Estos son consultados cuando se solicita un reporte y envían el response también en JSON |  | Solución temprana a la persistencia de las búsquedas tanto para historiales como para reportes | Conflicto con entrega 6 y 7.  necesidad de filtrar en memoria. | Guardarlas en una base de datos |
|  | Búsquedas almacenadas en medio relacional y no relacional |  | Tenemos las ventajas de ambos medios para usarlos según sea conveniente, Para los distintos reportes estadísticos del sistema se utiliza el origen relacional para recuperar la información básica de las búsquedas y tratarla más flexiblemente en memoria, mientras que para recuperar los historiales, el origen no relacional (Por requerimiento explícito del enunciado) y por la conveniencia de obtener tanto la información de las búsquedas como los poi resultados de las mismas de una forma directa. | Duplicamos el espacio utilizado | Elegir uno de los medios |
|  | Las acciones ante búsqueda se implementaron en un principio como propias del sistema. | Modelo Estático  Diagrama de Clases |  | Conflicto con la entrega 3 | que sean propias de cada usuario alterando solo el comportamiento individual de este. |
|  | La lógica de persistencia de las entidades se encapsuló en DAOs, | Modelo Estático | Simplifica las clases del modelo y aquellas que que necesitan una manipulación distinta pueden implementar un DAO específico. (Ejemplo son las búsquedas, donde la persistencia se realiza en dos medios). | Incremento en la cantidad de clases, la comunicación con la base debe atravesar otra capa. | Incluir la lógica de la base en el modelo |