Aplicación Java Sobre Web
Trabajo Integrador

INFORME

SPORTIA
Clases Deportivas y Recreativas

Fernando Pereyra

Contenido

1. INTRODUCCIÓN

1.1 PROPÓSITO DEL ENTREGABLE

Entregar una solución informática capaz de gestionar las inscripciones y pagos a clases de diferentes tipos (deportivas, recreativas, etc.) tanto de alumnos como de instructores.

1.2 RESUMEN EJECUTIVO DEL PROYECTO

Se pretende lograr el desarrollo de un software encargado de la gestión de inscripciones a clases que sea modular y altamente flexible, que permita automatizar el proceso de búsqueda, inscripción y pago de las mismas.

2. JUSTIFICACIÓN DEL PROYECTO

2.1 NECESIDADES DE NEGOCIO

- Lograr ventajas competitivas y mayores ganancias económicas.
- Acercar a los clientes los servicios brindados por la empresa.
- Dar mayor facilidad a las partes interesadas a adquirir nuestros productos/servicios.
- Promover el desarrollo de actividades que promuevan el bienestar de las personas.

2.2 OPORTUNIDADES

- Amplio mercado.
- Poca existencia de establecimientos recreativos en la zona.

3. ALCANCE

3.1 OBJETIVOS DEL PROYECTO

Construir un software modular y altamente flexible

Cumplir con los principios SOLID. Uso de patrones de diseño.

Facilitar la interacción de los interesados con nuestros servicios en lo que respecta a la búsqueda, inscripción y pago de las clases ofrecidas.

3.2 REQUISITOS DE ALTO NIVEL

Exponer las clases de la empresa y permitir filtrarlas por tipo, rango horario de inicio y zona en la cual se llevan a cabo.

Alta de usuarios con la debida autenticación y control de acceso.

Enviar un correo electrónico de notificación al usuario que se inscriba a una clase.

Realizar el cobro de la clase mediante un sandbox de pagos.

3.3 PRINCIPALES ENTREGABLES

- Página principal con registro e inicio de sesión disponibles y la posibilidad de ver las clases de SportIA.
- Búsqueda con filtro de clases. El filtro se debe poder hacer por tipo, lugar y horario.
- Inscripción a las clases con notificación por email correspondiente al usuario.
- Solución de pago mediante sandbox de pago.

Opcional:

- Integración con el bot de mensajería de Telegram.

4. FRAMEWORKS Y/O ARQUITECTURAS

La aplicación será web, con vistas a que también sea para dispositivos móviles, y escrita en Java. Para llevarla adelante se eligieron algunos marcos de trabajo que a continuación, se enumeran y describen, los cuales marcan las formas de desarrollo establecidas y las herramientas con las que se cuentan en la implementación de las diversas funcionalidades.

4.1 SPRING BOOT

Spring Boot está destinado a facilitar el desarrollo de aplicaciones basadas en Spring. Proporciona la función RAD (Rapid Aplicattion Development) lo que permite que el desarrollo de las aplicaciones sea más fluido y se pueda pensar más en la lógica de negocio y no tanto en la infraestructura. La idea de este proyecto es que se pueda comenzar con el mínimo esfuerzo, realizado una configuración mínima.

Spring Boot no genera código ni realiza ediciones en sus archivos. En su lugar, cuando inicia la aplicación, conecta dinámicamente los beans y la configuración, y los aplica al contexto de la aplicación.

Características

- Crea aplicaciones Spring independientes
- Se puede incrustar Tomcat, Jetty o Undertow directamente (no es necesario implementar archivos WAR)
- Proporciona dependencias "iniciales" obstinadas para simplificar su configuración de compilación
- Configura automáticamente Spring y bibliotecas de terceros siempre que sea posible
- Proporciona funciones listas para producción, como métricas, controles de estado y configuración externalizada
- Absolutamente sin generación de código y sin requisitos para la configuración XML

4.2 MAVEN

Apache Maven es una herramienta de comprensión y gestión de proyectos de software. Basado en el concepto de un modelo de objetos de proyecto (POM), Maven puede administrar la

construcción, los informes y la documentación de un proyecto desde una pieza central de información.

4.3 SRPING SECURITY

Este marco es utilizado para realizar dos funciones elementales de la seguridad de un sistema para las aplicaciones Java: la autenticación y el control de acceso. Spring Security permite personalizar fácilmente ambos procesos a la aplicación.

Características

- Soporte completo y extensible para autenticación y autorización
- Protección contra ataques como fijación de sesión, clickjacking, falsificación de solicitudes entre sitios (CSRF - cross site request forgery), etc.
- Integración de Servlet API
- Integración opcional con Spring Web MVC

4.4 THYMELEAF

Thymeleaf es un moderno motor de plantillas Java del lado del servidor para entornos web e independientes, capaz de procesar HTML, XML, JavaScript, CSS e incluso texto sin formato. Su objetivo es permitir la creación de plantillas de forma fácil y elegante. Fue diseñado teniendo en cuenta los estándares web, especialmente HTML5, lo que permite crear plantillas de validación completa si es necesario. Las plantillas HTML escritas en Thymeleaf se ven y funcionan como HTML. Presenta módulos específicamente diseñados para utilizarse con Spring Framework.

Presenta un dialecto estándar listo para usar, aunque se puede extender y personalizar la forma en que se procesan los datos en las plantillas, creando un dialecto que se adecue más con la aplicación.

En un proyecto Spring Boot la dependencia a agregar es la siguiente:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

5. BASE DE DATOS

5.1 MONGODB ATLAS

MongoDB Atlas es una plataforma de datos para aplicaciones multicloud. Posee un conjunto integrado de servicios de datos y bases de datos en la nube para acelerar y simplificar la gestión de los datos. Permite escalar las cargas de trabajo de la base de datos de manera segura y con alcance global.

En las relacionales SQL tradicionales existen tablas en dónde cada fila es un registro, en cambio, en MongoDB los datos se almacenan en colecciones (equivalente a las tablas) y documentos (equivalente a registros). Los documentos son del tipo BSON con pares de clave/valor como unidad básica de datos.

La base de datos MongoDB contiene colecciones que a su vez contienen documentos. La estructura del documento corresponde a la forma en la que se construyen las clases y objetos en la programación de la aplicación. Estos documentos no tienen un esquema predefinido y los campos pueden variar entre uno y otro, aunque pertenezcan a una misma colección.

MongoDB Atlas en este proyecto se usa en su versión gratuita que ofrece un clúster en AWS, Azure o Google Cloud en dónde se aloja la base de datos Mongo.

El nombre de la base de datos de la aplicación desarrollada es ***sportiadb*** y tiene los siguientes documentos: *users*, *lessons* e *inscriptions*.

5.2 NO SQL

El tipo de base de datos que se obtiene con MongoDB es NoSQL y orientada a documentos. La diferencia con las bases de datos relacionales es la flexibilidad y rendimiento que posibilita su diseño que como se mencionó no se requieren estructuras fijas. No necesitan ser normalizadas como sucede en un SGBDR (sistema de gestión de base de datos relacionales).

No soportan operaciones JOIN, ni aseguran que se cumplan las propiedades ACID (atomicidad, consistencia, aislamiento y durabilidad) pero tienen un buen escalamiento horizontal, agregando más servidores al clúster o más clústeres que se encarguen de la gestión de los datos en grandes volúmenes que pueden ser estructurados, semiestructurados y no estructurados. Este es el motivo por el cual se diseñaron, para poder hacer frente a la escalabilidad y agilidad que necesitan las aplicaciones modernas.

5.3 DER

users {
"_id": { ObjectId() },
"name": "",
"lastName": "",
"age": 0,
"dni": 0,
"email":"",
"password": “hash",
"phone": "",
"roles": [""],
"location": {
"country": "",
"state": "",
"city": "",
"code": "",
"street": "",
"number": 0
"floor": 0
"apartment": ""
},
"enabled": boolean,
"inscriptionsId": [""],
"_class":
"unaj.ajsw.sportia.model.User"
}

inscriptions {
"_id": { ObjectId() },
"lessonId": { ObjectId() },
"userId": { ObjectId() },
"payment": {
"status": "",
"external_reference": "",
"payment_type": "",
"merchant_order_id": "",
"preference_id": ""
},
"timestamp": {
"$date": {
"$numberLong":""
}
},
"_class":
"unaj.ajsw.sportia.model.Inscript
ion"
}

lessons {
"_id": { ObjectId() },
"name": "",
"type": "",
"weekDayNumber": 0,
"startTime": "",
"durationInMinutes": 0,
"location": {
"country": "",
"state": "",
"city": "",
"code": "",
"street": "",
"number": 0,
"floor": 0
"apartment": ""
},
"capacity": 0,
"price": 0.0,
"modality": "",
"purpose": "",
"active": boolean,
"enrolledId": [""],
"_class":"unaj.ajsw.sportia.mode
l.Lesson"
}

6. ASUNCIONES, RESTRICCIONES Y RIESGOS

6.1 ASUNCIONES

El equipo de desarrollo deberá comunicarse vía WhatsApp por cualquier consulta.

Uso de Software Jira para seguimiento de tareas.

Compartir un repositorio común en algún sistema de control de versiones.

Se pretende contar con los spikes necesarios que sean de utilidad para implementar el sandbox de pago y la integración con Telegram.

6.2 RESTRICCIONES

Poca experiencia en el desarrollo de aplicaciones con herramientas como *Spring Boot, Maven, Spring Security, Thymeleaf*.

6.3 RIESGOS

Al existir poca experiencia con el entorno se puede no lograr resolver un requerimiento en cuanto al desarrollo de la funcionalidad del sistema.

Abandono por parte de los miembros del equipo.

Tiempo limitado, no permitiendo un análisis más profundo de las herramientas y su implementación.

6.4 MEJORAS

- Agregar más métodos de pago.
- Agregar una especie de carrito de compras para realizar en una transacción la compra mensual de más de una clase.
- Implementar a través de api’s como WhatsApp o Telegram un bot que se encargue de guiar al usuario por la página web y de resolver dudas comunes.
- Implementar un espacio de intercambio entre usuarios del sistema para que puedan valorar y opinar sobre las clases de la empresa en las cuales se inscribieron.

7. ANEXOS

7.1 ANEXO I: DOCUMENTACIÓN DE REQUISITOS

En el Anexo I se definen en detalle los requisitos de la aplicación.

7.2 ANEXO II: MÉTODOS, MODELOS Y PRUEBAS

En el Anexo II se nombran y describen los métodos más importantes, implementados por servicio dentro de la aplicación. También se muestra el modelo utilizado para modelar los datos del sistema mediante un diagrama de clases. Por último, se describe brevemente algunos test realizados, propósitos y herramientas.