



Programa de Gestión Educativa

Chiessa, Leonardo Gabriel

Ramos, Marcos Ezequiel

Universidad Tecnológica Nacional Mar del Plata - 2020

Programa de Gestión Educativa

Índice

Programa de Gestión Educativa	2
Introducción	5
Definición del problema	5
Objetivos	6
Justificación	6
Marco Teórico	7
Tecnologías	7
HTTP	7
JSON	8
API REST	9
PROMESAS y OBSERVABLES	12
SPA	13
JWT	13
Herramientas y Lenguajes utilizados	15
JAVASCRIPT	15
JAVA PERSISTENCE API	15
MAVEN	16
SPRING BOOT	18
IntelliJ IDEA	19
ANGULAR	20
NODE.js	20
TYPESCRIPT	21
VISUAL STUDIO CODE	21
MySQL	22
MySQL Workbench	22

POSTMAN	24
HEROKU	25
TEXMAKER	26
Método	28
Proceso de Aprendizaje	28
Estado alcanzado	29
Recolección de datos mediante encuestas	30
Debilidades detectadas	32
Viabilidad	32
Revisión Preliminar	33
Antecedentes	33
Sobre el lenguaje empleado en el BackEnd	33
Sobre el lenguaje empleado en el FrontEnd	33
Sobre MySQL	33
Cronograma Product Backlog	33
Proceso de desarrollo	34
Puesta en marcha del entorno de trabajo	34
Spring 0	35
Spring 1	36
Spring 2	36
Spring 3	37
Spring 4	37
Planeamiento del método	39
Composición general tentativa o guion	39
Resultados	42
Documentación	42
Implementación	42
Pruebas	42

	4
Rendimiento	43
Conclusión	44
Semblanza de los candidatos	45
Chiessa, Leonardo Gabriel	45
Ramos, Marcos Ezequiel	45
Siglas	46
Glosario	47
Referencias	50

Introducción

Hoy en día la tecnología está presente en la vida cotidiana de todos los ciudadanos y estos interactúan con ella durante gran parte de su tiempo. Sin embargo podemos observar que este avance no llega a todos los organismos e instituciones sociales de la mejor manera. Ya que contribuye al acceso universal a la educación, vemos que a medida que se extienda su uso, tendrá un papel más importante en la enseñanza global.

En las Instituciones de educación pública de la ciudad de Mar del Plata no encontramos ningún software libre de gestión académica. Esto complica la tarea administrativa de forma considerable, teniendo en cuenta que pueden encontrarse más de mil alumnos matriculados por carrera por Institución, y cada una de ellas se arregla “como puede” según su presupuesto, que es menor al 10 % de lo recaudado por la Cooperadora Escolar anualmente.

Nuestra idea es romper con esto y brindar un software a medida según los lineamientos planteados en nuestra Tesina y enfrentarnos al desafío de profundizar los conocimientos adquiridos a lo largo de la carrera; estudiar y entender a fondo las tecnologías utilizadas, para ampliar la base de conocimiento para nuestro futuro como profesionales; devolver a la comunidad parte de lo que nos ha entregado, no solo a nivel educativo sino a nivel de aprendizaje, por lo tanto la herramienta a desarrollar será libre.

Definición del problema

Nuestro proyecto tiene como objetivo el desarrollo de un sistema para la gestión académica de los Institutos Terciarios de Mar del Plata. Con él, dichas instituciones tendrán una herramienta para llevar un control informatizado de los datos referentes a sus alumnos. Así los estudiantes podrán matricularse a las materias y mesas finales de forma online mediante el uso de Insufodo, dándole solución a un problema muy frecuente en el sistema educativo actual.

Se optó por este proyecto, debido a que un integrante del grupo actualmente trabaja en un Instituto Superior de Formación Docente estatal y conoce bien las problemáticas y la falta de preocupación que existe por parte del gobierno.

Objetivos

El proyecto fue desarrollado desde cero, siendo el objetivo final producir un software que gestione información sobre la situación académica de cada alumno (exámenes, cursadas, datos personales). Para que tanto estudiantes como empleados puedan realizar las tareas de índole administrativo en un espacio renovado.

Justificación

El desarrollo del proyecto surge de la necesidad de poder brindarle al equipo directivo del Instituto una herramienta clara y esencial para que el personal administrativo pueda cumplir sus tareas de manera eficiente. Se decidió relevar las alternativas a utilizar en este proyecto, de la parte de desarrollo de software así como también de las tecnologías aplicadas durante el proceso, mencionadas a continuación.

Marco Teórico

Tecnologías

HTTP. : Es un protocolo de comunicación de la capa de aplicación del Modelo OSI, que permite transferencias de información entre servidores y clientes en la WWW. Su primera versión fue la 0.9 que definía un protocolo simple para transferencia de datos crudos a través de Internet. Su versión 1.0 fue formalizada en el año 1996 a través del RFC 1945(m). Actualizado luego a su versión 1.1 en el año 1999 en el RFC 2616(n)[1] y mejorado en la familia RFC 723X(p-u) en el año 2014. Su última versión estable es la 2.0 definida en el año 2015 en el RFC 7540(w) documentados por la IETF. Es un protocolo de pedido/respuesta.

El cliente envía una solicitud al servidor especificando un método de pedido, URI y versión del protocolo, seguido por un mensaje MIME que contiene modificadores de solicitud, información del cliente y, opcionalmente, un cuerpo con contenido.

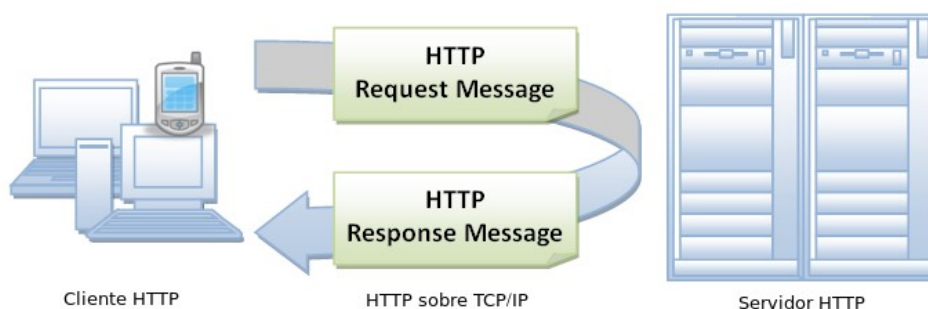


Figura 1. Gráfico #1: estructura de una comunicación Http

El gráfico #2 muestra la estructura de una solicitud HTTP. El servidor responde con un mensaje que incluye una línea de estado, que contiene la versión del protocolo y un código de estado, seguido por un mensaje MIME con información del servidor, meta-información de la entidad y, opcionalmente, un cuerpo con contenido. El gráfico #3 muestra la estructura de una respuesta Http.

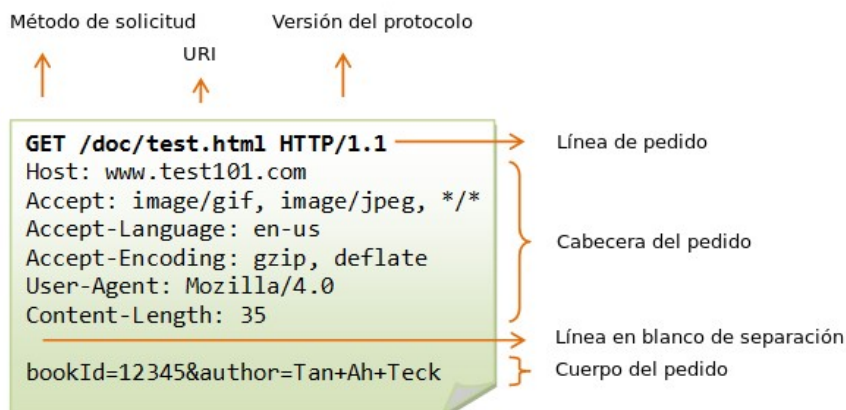


Figura 2. Gráfico #2: estructura de una petición HTTP.

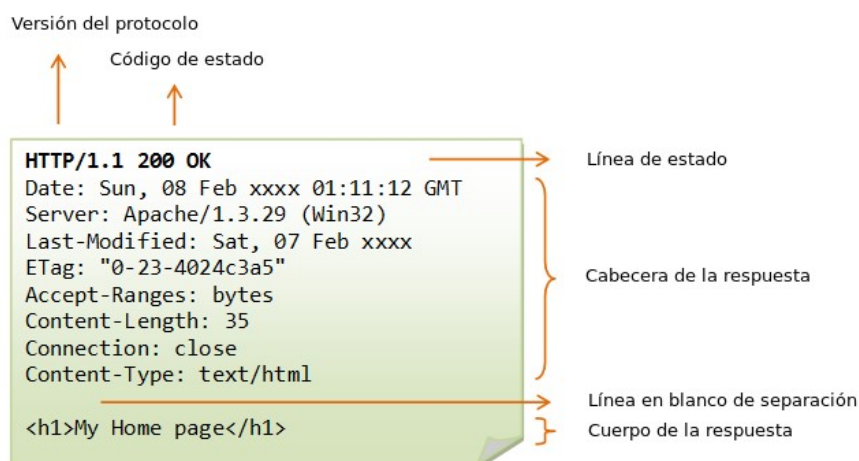


Figura 3. Gráfico #3: estructura de una respuesta HTTP.

JSON. : Es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros.

Estas propiedades hacen que sea un lenguaje ideal para el intercambio de datos. JavaScript Object Notation (JSON) está constituido por dos estructuras:

1. Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como

un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.

2. Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras. En JSON, se presentan de estas formas:

```
{
  "idAsignatura": {
    "idAsignatura": 1,
    "nombreAsignatura": "Cálculo"
  },
  "idEstudiante": {
    "idEstudiante": 1,
    "idUsuario": {
      "emailUsuario": "victoria@sawp.cl",
      "idUsuario": 3,
      "nombreUsuario": "Victoria Martínez",
      "pwdUsuario": "1234",
      "rolUsuario": "Estudiante"
    },
    "matriculado": 1
  },
  "idTareas": 1,
  "nombreTarea": "Series de Fourier",
  "tareaNota": 7
},
```

Figura 4. **Gráfico #4:** estructura de un objeto JSON.

Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con llave de apertura y termine con llave de cierre. Cada nombre es seguido por: dos puntos y los pares nombre/valor están separados por coma.[2]

API REST. : **API** es una capa de abstracción la cual consiste en un conjunto de métodos y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software.

REST es un estilo de arquitectura de desarrollo de software para sistemas distribuidos como la WWW. Fue originalmente definido en el año 2000 por Roy Fielding(a)[3], coautor de la especificación HTML. Define siete propiedades arquitectónicas:

- Rendimiento.
- Escalabilidad.
- Simplicidad.
- Modificabilidad.
- Visibilidad.
- Portabilidad.
- Confiabilidad.

y una serie de restricciones para guiar a un sistema a cumplirlas. Si un servicio viola alguna de las restricciones, no puede ser considerado **RESTfull**:

- Cliente - Servidor: la arquitectura de la aplicación debe ser del tipo cliente-servidor. Este tipo de arquitectura permite una separación de intereses, mejorando la portabilidad de la interfaz de usuario a través de múltiples plataformas y la escalabilidad, simplificando los componentes del servidor.
- Sin estado: la solicitud HTTP del cliente contiene toda la información necesaria para comprender la petición y el estado de la sesión debe mantenerse en el cliente. Este estado puede ser transferido por el servidor a otro servicio, como una base de datos, para mantener una ventana de trabajo y permitir autenticación.
- Cacheable: las respuestas HTTP deben, implícita o explícitamente definirse como cacheables o no, para prevenir que los clientes reutilicen información inapropiada en peticiones futuras.
- Sistema en capas: un cliente no puede indicar si está conectado directamente al servidor final o a un intermediario. Los servidores intermediarios pueden mejorar la escalabilidad permitiendo balanceo de cargas o la seguridad, forzando políticas de seguridad.

- Código bajo demanda: los servidores pueden extender temporalmente la funcionalidad de un cliente transfiriendo código ejecutable, como Applets de Java o Script de JavaScript.
- Interfaz uniforme: definida en cuatro sub-restricciones:
 - Identificación de recursos: cada recurso individual es identificado en una petición por su URL. Las representaciones de estos recursos que devuelve el servidor están separadas conceptualmente de la representación interna que este posee.
 - Manipulación de los recursos a través de representaciones: cuando un cliente tiene una representación de un recurso, posee la información necesaria para modificarlo o eliminarlo.
 - Mensajes auto-descriptivos: cada mensaje incluye información suficiente para describir cómo se procesa el mensaje.
 - HATEOAS (Hypermedia as the engine of application state): habiendo accedido a una URI inicial para una aplicación REST, un cliente debería poder acceder dinámicamente a través de enlaces provistos por el servidor a cualquier recurso disponible en el servidor.

Para manipular los recursos, HTTP nos dota de los siguientes métodos con los cuales debemos operar:

- GET: Para consultar y leer recursos
- POST: Para crear recursos
- PUT: Para editar recursos
- DELETE: Para eliminar recursos.
- PATCH: Para editar partes concretas de un recurso.

PROMESAS y OBSERVABLES. : Una **promesa** de JavaScript es la asignación de un valor asincrónicamente, es decir se asignará en un futuro. Las promesas empezaron en DOM como "Future", se les cambió el nombre a "Promises", finalmente, se trasladaron a JavaScript. Es fabuloso contar con ellas en JavaScript en lugar de en el Dom, porque estarán disponibles en contextos de JS sin navegador, como Node.js. Si bien son una funcionalidad de JavaScript, el DOM las usa sin problemas cuando las necesita. De hecho, todas las nuevas APIs de DOM con métodos de éxito o falla asincrónicos usan promesas. Esto ya ocurre en la administración de cuotas, los eventos de carga de fuentes, ServiceWorker, Web MIDI, las transmisiones y mucho más.

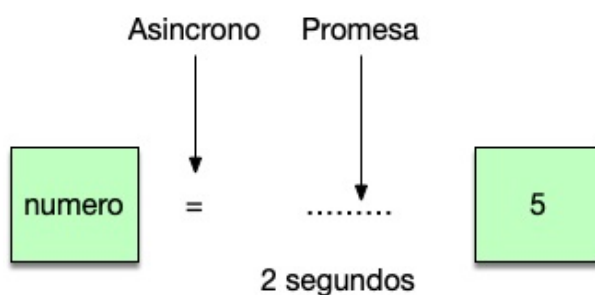


Figura 5. **Gráfico #5:** Promesas asincronicas

Un **observable** no es ni más ni menos que una colección de elementos asíncronos. Cada vez que pulsamos un botón en un interface de usuario se produce un evento de click. Se trata de una colección de elementos asíncronos.

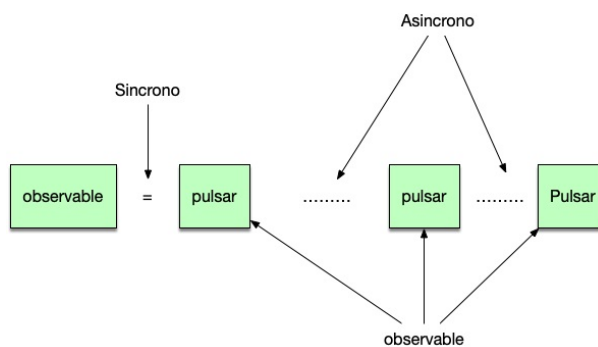


Figura 6. **Gráfico #6:** Observable.

SPA. : Single Page Application es una aplicación web de una sola página diseñada con el propósito de dar una experiencia similar al de una aplicación de escritorio. En ellas, se carga todo el contenido HTTP, CSS y JavaScript necesario para representar el estado inicial de la aplicación y pueden obtenerse recursos adicionales a través de peticiones sucesivas al servidor (mediante Promesas), brindando una experiencia de usuario más fluida. Una SPA no recarga en ningún punto del proceso la página mostrada ni transfiere el control a otra página, aunque pueden utilizarse ciertos mecanismos, como la API de historial de HTML5, para proveer la percepción de navegabilidad entre páginas lógicas.

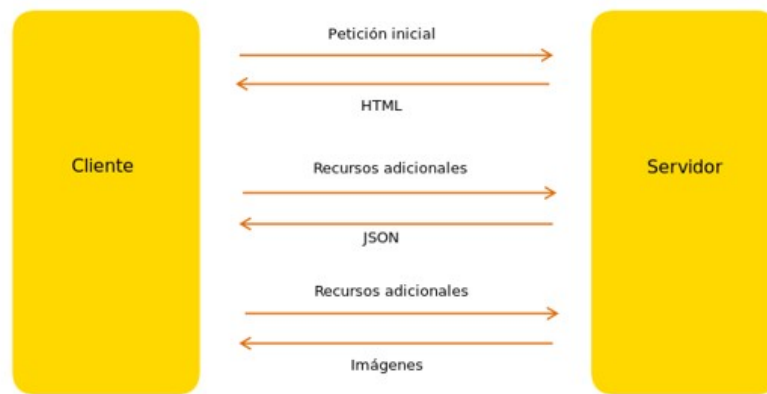


Figura 7. Gráfico #7: flujo de comunicaciones de una SPA

JWT. : del inglés JSON Web Token es un estándar abierto especificado a través del RFC 7519(v)[4] que define una forma compacta y autocontenida para transmitir información asegurizada entre partes en forma de un objeto JSON. Esta información está firmada digitalmente, por lo que puede confiarse en su contenido.

Los JWT consisten de tres partes separadas por un punto, las cuales son:

- **Cabecera:** formada típicamente por dos partes. El tipo de token (JWT) y el algoritmo utilizado en la firma digital (HMAC o RSA), los cuales luego se codifican utilizando Base64Url.
- **Carga útil:** es la segunda parte de un JWT, la cual está subdividida en tres tipos de propiedades:

```

1 {
2   "alg": "HS256",
3   "typ": "JWT"
4 }

```

Figura 8. Gráfico #8: ejemplo de una cabecera JWT.

- Propiedades reservadas: un conjunto de propiedades que no son obligatorias pero sí recomendadas. Por ejemplo, iss (del inglés issuer) quien expide el token; exp (del inglés expiration time) tiempo de duración del token, sub (del inglés subject) motivo del token, etc.
- Propiedades públicas: definidas por cualquiera que utilice JWT, pero deben ser registradas en el Registro de Web Token JSON de la IANA para evitar colisiones.
- Propiedades privadas: propiedades personalizadas para compartir información entre partes utilizando los JWT.

```

1 {
2   "sub": "1234567890",
3   "name": "John Doe",
4   "admin": true
5 }

```

Figura 9. Gráfico #9: ejemplo de carga útil JWT.

- Firma: para crear la firma se toman la cabecera codificada y la carga útil codificada y se firman. Por ejemplo, si se desea utilizar el algoritmo HMAC SHA256 la firma se crearía de la siguiente forma:

```

1 HMACSHA256(
2   base64UrlEncode(header) + "." +
3   base64UrlEncode(payload),
4   secret)

```

Figura 10. Gráfico #10: ejemplo de codificación de una firma JWT.

La salida es una cadena de texto Base64 como la siguiente:

```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4.
gRG9lIiwiaXNTb2NpYWwiOiJhbnRydWV9.
4pcPyMD09o1PSyXnrXCjTwXyr4BsezdI1AVTmud2fU4

```

Figura 11. Gráfico #11: estructura final de un token JWT.

Herramientas y Lenguajes utilizados

JAVASCRIPT. : Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas y JavaScript del lado del servidor (Server-side JavaScript o SSJS)[5].

- Lenguaje de alto nivel.
- Son independientes de la arquitectura de la maquina, está dirigido a software, por lo que puede migrarse y utilizarse entre diferentes sistemas operativos.
- Es uno de los lenguajes más usados del mundo, su función principal es animar las páginas web, y se aplica a los documentos Html.

JAVA PERSISTENCE API. : Java Persistence API (JPA) en español persistencia desarrollada para Java. Es un Framework que mapea los datos de los modelos de una aplicación en entidades relacionales. La persistencia en este contexto cubre tres áreas:

- La API en sí misma, definida en el paquete javax.persistence.
- El lenguaje de consulta Java Persistence Query Language (JPQL).
- Metadatos objeto/relacional.

El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos (siguiendo el patrón de mapeo objeto-relacional). El mapeo objeto-relacional es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo)[6].

MAVEN. : Una de las herramientas más útiles a la hora de utilizar librerías de terceros es Maven. Se utiliza en la gestión y construcción de software. Posee la capacidad de realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. Es decir, hace posible la creación de software con dependencias incluidas dentro de la estructura del JAR. Es necesario definir todas las dependencias del proyecto (librerías externas utilizadas) en un fichero propio de todo proyecto, el Project Object Model (POM). Este es un archivo en formato XML que contiene todo lo necesario para que, a la hora de generar el fichero ejecutable de nuestra aplicación, este contenga todo lo que necesita para la ejecución en su interior.

La característica más importante de Maven es su capacidad de trabajar en red. Cuando definimos las dependencias, este se encargará de ubicar las librerías que deseamos utilizar en Maven Central, el cual es un repositorio que contiene cientos de librerías constantemente actualizadas por sus creadores. Maven permite incluso buscar versiones más recientes o más antiguas de un código dado y agregarlas a nuestro proyecto. Todo se hará de forma automática sin que el usuario tenga que hacer nada más que definir las dependencias.

- Se basa en patrones y en estándares. Esto permite a los desarrolladores moverse entre proyectos y no necesitan aprender como compilar o empaquetar. Esto mejora el mantenimiento y la reusabilidad.
- Hace la gestión de librerías, incluso teniendo en cuenta las dependencias transitivas. Es decir, si A depende de B y B depende de C, es que A depende de C. Esto quiere decir que cuando empaquetemos A, Maven se encargará de añadir tanto B como C en el paquete.

Maven utiliza un ciclo de vida de construcción para automatizar varias fases de nuestro proyecto: Compilación, Pruebas Unitarias, Empaquetado, Pruebas de integración, verificaciones adicionales y distribución del empaquetado resultante hacia un repositorio local y/o centralizado. Cuando lo instalamos crea automáticamente el Repositorio Local, el mismo servirá como un punto de cache entre los servidores Maven de internet y nuestra

estación de trabajo[7]. Contiene la información que hace único al artefacto que el proyecto genera, la información necesaria es:

- **groupId** Identificación del grupo al que pertenece el proyecto.
- **id** Nombre de nuestro proyecto.
- **packaging** Tipo de empaquetado (por ejemplo JAR o WAR).

Dependencias: Se definen en el POM y constituyen un vínculo hacia las librerías de terceros utilizadas por el proyecto y los mecanismos de publicación.

Lombok: Cuando programamos una clase, nos encontramos con la tarea de escribir los métodos Setter, Getter, Constructores, etc. Lombok es una librería que genera ese código repetitivo por nosotros. No de la forma que lo hacen los IDE como Eclipse o IntelliJ IDEA dentro del archivo .java, sino dentro del archivo compilado .class. Logrando un código más limpio y de fácil mantenimiento, en el que por medio de anotaciones vamos definiendo lo que necesitamos hacer.

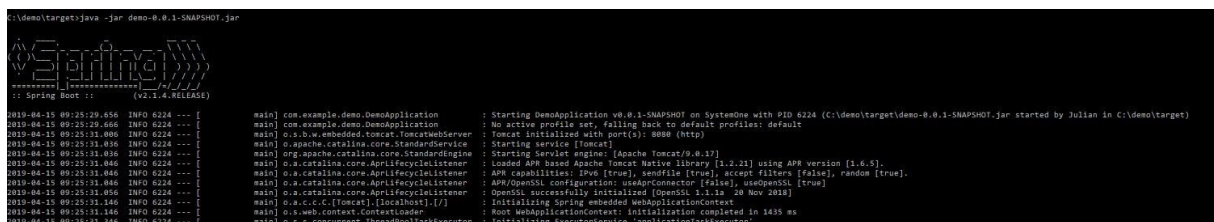
Model Mapper: Las aplicaciones requieren objetos que aunque son muy similares en cuanto a estructura/necesidades tienen entre ellos leves modificaciones. El mapeo tiene la finalidad de facilitar la conversión, el traspaso de los datos de un objeto entity a otro. El objetivo de Model Mapper es hacernos el mapeo entre objetos más sencillo. Algunos de los beneficios de utilizar Model Mapper son:

- Basado en convenciones e inteligente: no necesita un mapeo manual (a excepción de los campos cuyos nombres no coincidan tanto en nombre como en tipo de dato).
- API simple y segura: al no tipear nosotros el código manualmente y hacerse automáticamente por detrás evitamos la posibilidad de que se produzcan posibles fallos.
- Extensible: admite diferentes formatos de modelos de datos. JavaBeans, JSON, BBDD.

SPRING BOOT. : Publicado en 2012, es una solución para el framework Spring de Java que sigue el principio de “convención sobre configuración” y reduce la complejidad del desarrollo de nuevos proyectos basados en Spring. Para ello, proporciona la estructura básica configurada del proyecto, que incluye las pautas para usar el marco y todas las bibliotecas de terceros relevantes para la aplicación, lo que nos allana el camino para comenzar a desarrollarla lo más rápidamente posible. De esta manera se simplifica mucho la creación de aplicaciones independientes y reproducibles, por lo que la mayoría de las nuevas aplicaciones basadas en Spring se desarrollan con Spring Boot. Las características principales pueden resumirse de la siguiente manera:

- **Incorporación** directa de aplicaciones de servidores web/contenedores como Apache Tomcat o Jetty, eliminando la necesidad de incluir archivos Web Application Archive (WAR).
- **Simplificación** de la configuración de Maven gracias a los POM “starter”.
- **Configuración** automática de **Spring** en la medida de lo posible.
- Características no funcionales, como métricas o configuraciones externalizadas.

La empresa desarrolladora Pivotal Software dio al framework de Spring[13], publicado ya en 2005, un enfoque moderno y con visión de futuro cuando lanzó Spring Boot. Para desarrollarlo, se recurrió a la sólida tecnología de base del framework de Spring, madurada durante años, que contribuyó mucho a su funcionalidad.



```

C:\demo\target>java -jar demo-0.0.1-SNAPSHOT.jar

Spring Boot 2.1.4.BUILD-SNAPSHOT

2019-04-15 09:25:29.666 INFO 6224 --- [main] com.example.demo.DemoApplication : Starting DemoApplication v0.0.1-SNAPSHOT on SystemDns with PID 6224 (C:\demo\target\demo-0.0.1-SNAPSHOT.jar started by Julian in C:\demo\target)
2019-04-15 09:25:29.666 INFO 6224 --- [main] com.example.demo.DemoApplication : No active profile set, falling back to default profiles: default
2019-04-15 09:25:31.080 INFO 6224 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8088 (http)
2019-04-15 09:25:31.096 INFO 6224 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.17]
2019-04-15 09:25:31.096 INFO 6224 --- [main] org.apache.catalina.core.StandardEngine : Starting service [Tomcat]
2019-04-15 09:25:31.096 INFO 6224 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.17]
2019-04-15 09:25:31.096 INFO 6224 --- [main] o.a.catalina.core.AprLifecycleListener : Loaded APR based Apache Tomcat Native library [1.2.21] using APR version [1.6.5].
2019-04-15 09:25:31.096 INFO 6224 --- [main] o.a.catalina.core.AprLifecycleListener : APR capabilities: IPv6 [true], sendfile [true], accept filters [false], random [true].
2019-04-15 09:25:31.096 INFO 6224 --- [main] o.a.catalina.core.AprLifecycleListener : APR/OpenSSL configuration: useAprConnector [false], useOpenSSL [true]
2019-04-15 09:25:31.096 INFO 6224 --- [main] o.a.catalina.core.AprLifecycleListener : OpenSSL successfully initialized (OpenSSL 1.1.1g 20 Nov 2018)
2019-04-15 09:25:31.140 INFO 6224 --- [main] o.a.c.c.c.[Tomcat].localhost [/] : Initializing Spring embedded WebApplicationContext
2019-04-15 09:25:31.140 INFO 6224 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1435 ms
2019-04-15 09:25:31.140 INFO 6224 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
  
```

Figura 12. Gráfico #12: Inicio de la aplicación en Spring Boot.

IntelliJ IDEA. : Entorno de desarrollo para la codificación del desarrollo lógico del BackEnd. Cada aspecto de esta IDE diseñada para maximizar la productividad del desarrollador. En conjunto, la asistencia de codificación inteligente y el diseño ergonómico hacen que el desarrollo no solo sea productivo sino también agradable.

A continuación se nombran los puntos mas fuertes:

- **Inteligencia profunda:** Después de que IntelliJ IDEA haya indexado su código fuente, ofrece una experiencia inteligente y ultrarápida aportando sugerencias relevantes en cada contexto: finalización de código instantánea e inteligente, análisis del código al momento y herramientas de refactorización fiables.
- **Experiencia lista para usar:** Herramientas esenciales como sistemas de control de versiones integrados y una amplia variedad de lenguajes compatibles y marcos de trabajo están todas integradas y a su alcance, sin necesidad de complicados complementos.
- **Finalización inteligente de código:** Mientras que la finalización básica sugiere nombres de clases, métodos, campos y palabras clave en el ámbito de la visibilidad, la finalización inteligente sugiere solo aquellos tipos que se esperan en el contexto actual.
- **Asistencia específica para el marco de trabajo:** Mientras que IntelliJ IDEA es un IDE para Java, también entiende y ofrece asistencia de codificación inteligente para una gran variedad de otros lenguajes como SQL, JPQL, HTML, JavaScript, etc., incluso si la expresión del lenguaje está inyectada en una cadena literal en su código Java.
- **Aceleradores de productividad:** El IDE predice sus necesidades y automatiza las tediosas y repetitivas tareas de desarrollo, de modo que usted puede centrarse en lo importante.
- **Ergonomía del desarrollador:** En cada decisión sobre la implementación y el diseño que tomamos, tenemos en cuenta el riesgo de interrumpir el flujo del desa-

rollador y hacer todo lo posible para evitarlo o minimizarlo. El IDE sigue el contexto y sugiere las herramientas correspondientes de manera automática.

- **Inteligencia discreta:** La asistencia a la codificación en IntelliJ IDEA no incluye solo al editor: le ayuda a mantener la productividad cuando trata también otros aspectos como, por ejemplo, rellenar un campo, buscar en una lista de elementos, acceder a la ventana de una herramienta o cambiar un ajuste, etc.

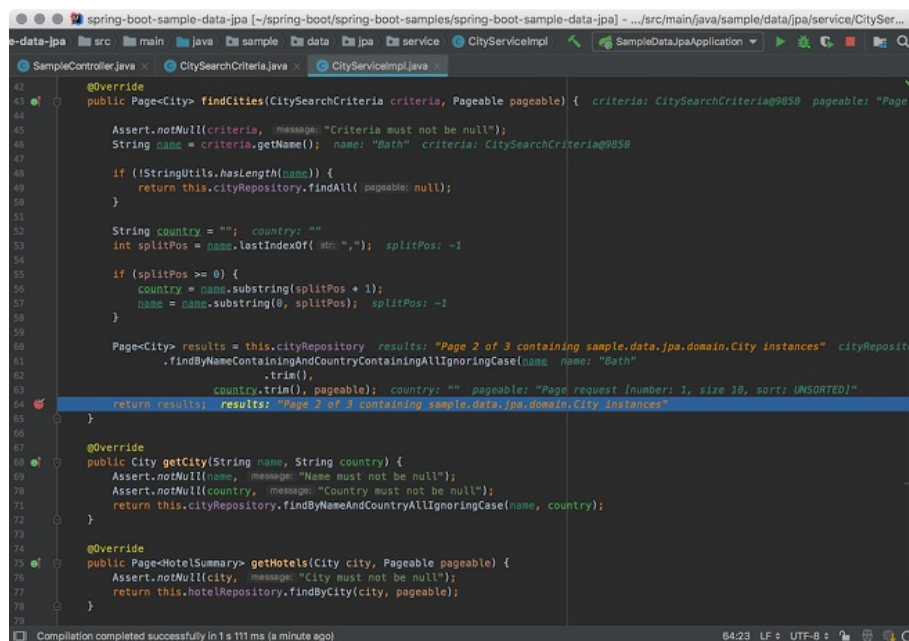


Figura 13. Gráfico #13: Captura de pantalla de código de una aplicación Spring Boot.

ANGULAR. : Se utiliza en el FrontEnd, en su versión 8 es la plataforma perfecta para el desarrollo profesional de aplicaciones modernas. El Angular CLI es la herramienta de línea de comandos adecuada para crear, depurar y publicar aplicaciones Angular. Juntos son imbatibles en cuanto a velocidad en desarrollo y potencia en ejecución[14].

NODE.js. : Entorno de ejecución para el desarrollo de una variedad de herramientas y aplicaciones . Aunque no es un framework JavaScript, muchos de sus módulos básicos están escritos en JavaScript y los desarrolladores pueden también escribir nuevos módulos con él. Tiene una arquitectura basada en eventos y está enfocado en la optimización de escalabilidad y rendimiento para aplicaciones web que utilicen un fuerte procesamiento de entrada/salida asíncrono. Desarrollado inicialmente por Ryan Dahl[8] en el año 2009, se encuentra actualmente en la versión 12.15.0 (incluye npm 6.13.4).

TYPESCRIPT. : Es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases. Puede ser usado para desarrollar aplicaciones que se ejecutarán en el lado del cliente o del servidor (Node.js). Es llamado también Superset (lenguaje escrito sobre otro lenguaje), lo que significa que si el navegador está basado en Javascript, este nunca llegará a saber que el código original fue realizado con TypeScript[9]. Angular uno de los frameworks más famosos de TypeScript.

VISUAL STUDIO CODE. : Entorno de desarrollo para la codificación del desarrollo lógico del FrontEnd. Es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C ++, C#, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity).

Se basa en Electron, un framework que se utiliza para implementar aplicaciones Node.js para el escritorio, que se ejecuta en el motor de diseño Blink. Aunque utiliza Electron, emplea el mismo componente editor (Monaco) utilizado en Visual Studio Team Services (anteriormente llamado Visual Studio Online). Es gratuito y de código abierto [10].

A continuación se nombran los puntos mas fuertes:

- **Finalización inteligente de código:** Código más inteligente con IntelliSense para terminaciones para variables, métodos y módulos importados.
- **Depuración simplificada:** Excelente soporte de depuración. El depurador incorporado (Node.js) de VS Code ayuda a acelerar su ciclo de edición, compilación y depuración.
- **Edición rápida y potente:** Linting (proporciona advertencias para códigos sospechosos) , edición con varios cursores , sugerencias de parámetros y otras potentes funciones de edición.
- **Código de navegación y refactorización:** Posibilidad de explorar su código fuente rápidamente por referencias y navegue hasta la definición .

- **Control de fuente en el producto:** Integrado el control de origen e incluye compatibilidad con Git in-the-box. Muchos otros proveedores de control de fuente están disponibles a través de extensiones en VS Code Marketplace.

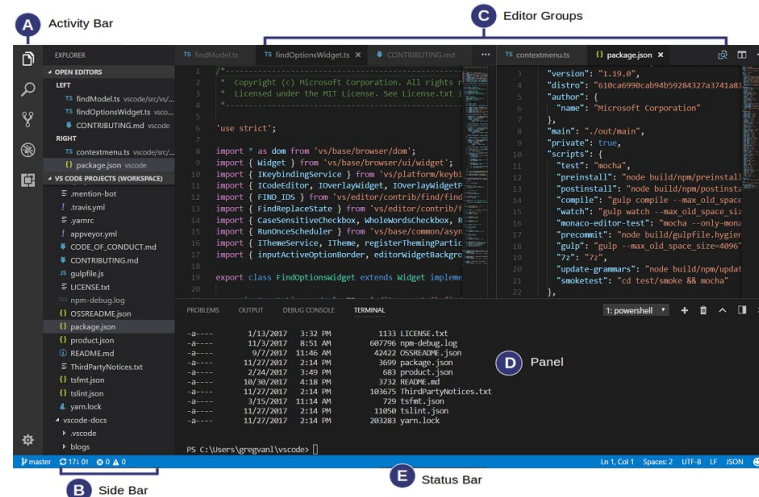


Figura 14. Gráfico #14: Captura de pantalla de código en VS Code.

MySQL. : Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base de datos open source más popular del mundo. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, en aplicaciones web cuando hay baja concurrencia en la modificación de datos y, en cambio, cuando el entorno es intensivo en lectura de datos. Esto hace a MySQL ideal para este tipo de aplicaciones.

MySQL Workbench. : Especialmente orientada al diseño y modelado de bases de datos MySQL. Para ello podemos crear los diagramas DER de manera detallada, pudiendo especificar plenamente las características de tablas, campos, claves, relaciones entre tablas, etc. Es simple añadir y editar posteriormente tablas y relaciones haciendo uso de los botones de la barra de herramientas y los menús correspondientes.

Para las tablas es posible especificar, por ejemplo, el motor de la base de datos, el lenguaje, los índices, campos, claves foráneas, incluso triggers o privilegios. Además de poder personalizar al completo tanto la notación en los diagramas DER como la propia interfaz gráfica de la aplicación, podemos realizar operaciones de gran importancia.

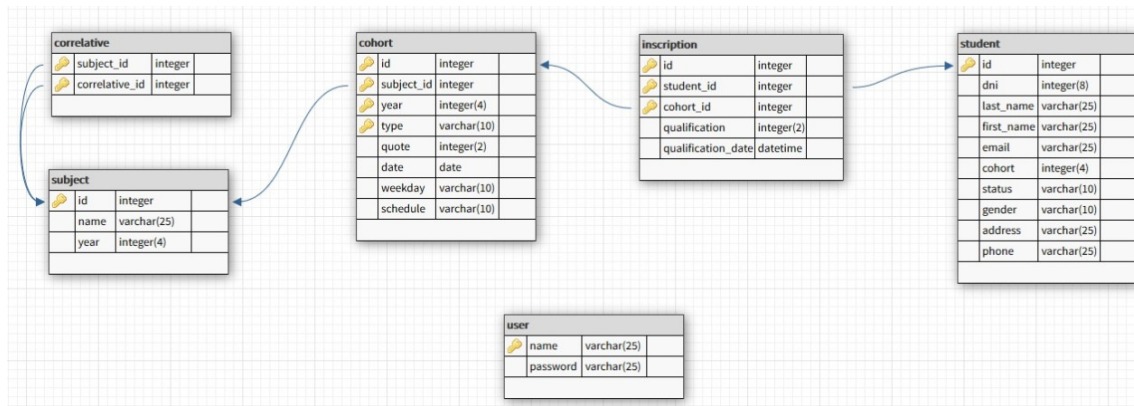


Figura 15. Gráfico #15: DER Insufodo.

MySQL Workbench se basa en cinco puntos importantes los cuales son los siguientes:

1. **Desarrollo de SQL:** Permite crear y administrar conexiones a servidores de bases de datos. Además de permitirle al usuario el poder configurar los parámetros de conexión, MySQL Workbench brinda la capacidad de ejecutar consultas SQL en las conexiones de la base de datos utilizando el Editor de SQL incorporado.
2. **Modelado de datos (diseño):** permite crear modelos del esquema de su base de datos de forma gráfica, aplicar ingeniería inversa y directa entre un esquema y una base de datos activa, y editar todos los aspectos de su base de datos utilizando el completo Editor de tablas. El editor de tablas proporciona facilidades de fácil uso para editar tablas, columnas, índices, disparadores, particiones, opciones, inserciones y privilegios, rutinas y vistas.
3. **Administración del servidor:** permite administrar instancias del servidor MySQL al administrar usuarios, realizar copias de seguridad y recuperación, inspeccionar datos de auditoría, ver el estado de la base de datos y monitorear el rendimiento del servidor MySQL.
4. **Migración de datos:** permite migrar de Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostgreSQL y otras tablas, objetos y datos RDBMS a MySQL. La migración también admite la migración de versiones anteriores de MySQL a las últimas versiones.

5. MySQL Enterprise Support: soporte para productos empresariales como MySQL Enterprise Backup, MySQL Firewall y MySQL Audit.

MySQL Workbench también puede generar el script necesario para crear la base de datos que se ha dibujado en el esquema; es compatible con los modelos de base de datos de DBDesigner 4 y soporta las novedades incorporadas en MySQL 5.

POSTMAN. : Se utiliza, sobre todo, para el testing de API REST , aunque también admite otras funcionalidades que se salen de lo que engloba el testing de este tipo de sistemas. Gracias a esta herramienta, además de testear, consumir y depurar API REST, podremos monitorizarlas, escribir pruebas automatizadas para ellas, documentarlas, mockearlas, simularlas, etc.

Está compuesto por diferentes herramientas y utilidades gratuitas (en la versión free) que permiten realizar tareas diferentes dentro del mundo API REST: creación de peticiones a APIs internas o de terceros, elaboración de tests para validar el comportamiento de APIs, posibilidad de crear entornos de trabajo diferentes (con variables globales y locales), y todo ello con la posibilidad de ser compartido con otros compañeros del equipo de manera gratuita (exportación de toda esta información mediante URL en formato JSON).

El interés fundamental de Postman es utilizarlo como una herramienta para hacer peticiones a APIs y generar colecciones de peticiones que nos permitan probarlas de una manera rápida y sencilla.

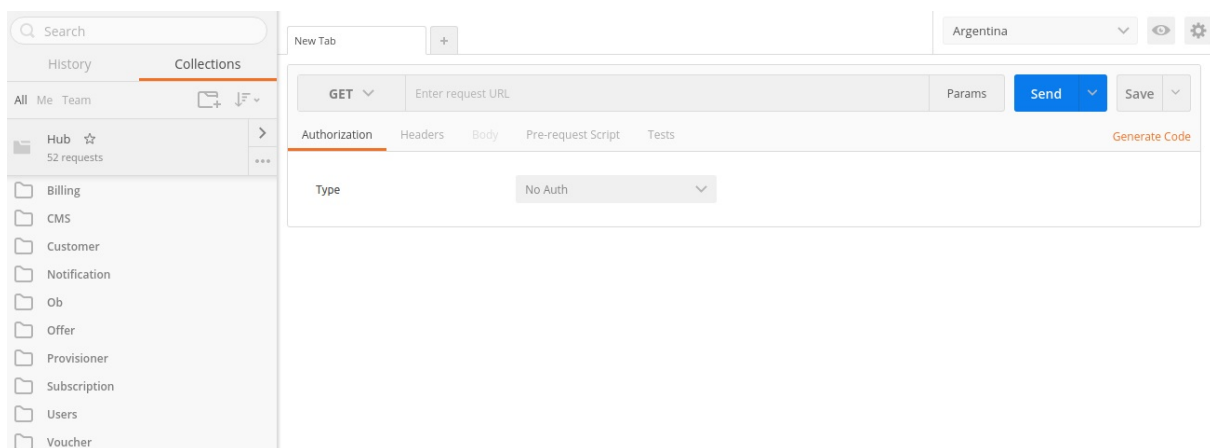


Figura 16. Gráfico #16: Captura de pantalla del entorno Postman.

Como vemos, se puede definir el tipo de petición (GET, POST, etc.), tokens de autenticación, cabeceras asociadas a la petición, etc., todo de una manera muy sencilla. Todas las llamadas almacenadas en nuestra colección pueden ser exportadas a múltiples lenguajes haciendo clic en el apartado Generate Code que se puede observa en el gráfico #16.

HEROKU. : Desde el lado del Deploy encontramos esta plataforma que posibilito subir a la nube de Internet nuestra SPA Landing Page de alumnos al siguiente enlace <https://insufodo.herokuapp.com/>.

Funciona como un servicio basado en un sistema contenedor administrado, con servicios de datos integrados y un poderoso ecosistema, para implementar y ejecutar aplicaciones modernas. La experiencia de desarrollador de Heroku es un enfoque centrado en la aplicación para la entrega de software, integrado con las herramientas y flujos de trabajo de desarrollador más populares de la actualidad.

Sus principales características son :

- **Runtime:** Ejecuta sus aplicaciones dentro de dynos : Contenedores inteligentes en un entorno de tiempo de ejecución confiable y totalmente administrado . Los desarrolladores implementan su código escrito en Node , Ruby , Java , PHP , Python , Go , Scala o Clojure en un sistema de compilación que produce una aplicación que está lista para su ejecución. El sistema y las pilas de idiomas se monitorean, revisan y actualizan, por lo que siempre está listo y actualizado. El tiempo de ejecución mantiene las aplicaciones ejecutándose sin ninguna intervención manual.
- **Experiencia de desarrollador (DX):** Es un enfoque centrado en la aplicación a la entrega de software para que los desarrolladores pueden concentrarse en la creación y la entrega de forma continua las aplicaciones, sin ser distraído por los servidores o infraestructura. Los desarrolladores implementan directamente desde herramientas populares como los sistemas Git, GitHub o Continuous Integration (CI). El panel de control intuitivo Heroku hace que sea fácil administrar su aplicación y obtener una mayor visibilidad del rendimiento.

- **Servicios de datos y ecosistema:** Permite a los desarrolladores ampliar sus aplicaciones con complementos, personalizar su pila de aplicaciones con Buildpacks y comenzar sus proyectos con botones. Los complementos son servicios en la nube de terceros que los desarrolladores pueden usar para extender inmediatamente sus aplicaciones con una variedad de funcionalidades, como almacenes de datos, registro, monitoreo y más.
- **Experiencia operacional (OpEx):** Componente clave de la plataforma. Ayuda a los desarrolladores a resolver problemas y solucionar problemas comunes y a personalizar su experiencia de operaciones para identificar y abordar rápidamente las tendencias negativas en el estado de sus aplicaciones.
- **Seguridad y cumplimiento:** Los desarrolladores de todo el mundo confían datos confidenciales a Heroku, y nada es más importante que proteger estos datos. Heroku realiza auditorías regularmente y mantiene el cumplimiento de PCI, HIPAA, ISO y SOC.

TEXMAKER. : Es un editor gratuito distribuido bajo la licencia GPL para escribir documentos de texto, multiplataforma, que integra muchas herramientas necesarias para desarrollar documentos con LaTeX, en una sola aplicación. Texmaker incluye soporte Unicode, corrección ortográfica, auto-completado, plegado de código y un visor incorporado en pdf con soporte de syntex y el modo de visualización continua.

LaTeX es un sistema de preparación de documentos. Con él puedes preparar manuscritos, artículos de revista, cartas, tesis, presentaciones y cualquier tipo de documento que quisieras imprimir en papel o mostrar en pantalla.

En la superficie, una de las ventajas de LaTeX es la calidad profesional de los documentos que puedas generar. Esto es particularmente cierto para documentos que contengan fórmulas o ecuaciones, tiene muchas aplicaciones más allá de las matemáticas. Documentos de química, física, computación, biología, leyes, literatura, música y cualquier otro tema que se puedan ocurrir, pueden igual aprovechar la excelente calidad de imprenta.

Una ventaja menos obvia, pero quizá más importante, es que LaTeX te permite claramente separar el contenido y el formato de tu documento. Como científico, investigador o escritor, esto te da la oportunidad de concentrarte en el ‘qué’, en la parte creativa de tu obra, en generar y escribir ideas. Por su parte el sistema se encargará del ‘cómo’ hacer para plasmar esas ideas visualmente en un documento. LaTeX, además, realiza de manera automática muchas tareas que de otro modo podrían resultar tediosas o engorrosas: numerar capítulos y figuras, incluir y organizar la bibliografía adecuada, mantener índices y referencias cruzadas.

Finalmente, lo que para algunos es también un punto a favor, todo el software que necesitas para editar, producir, ver e imprimir tus documentos es libre.

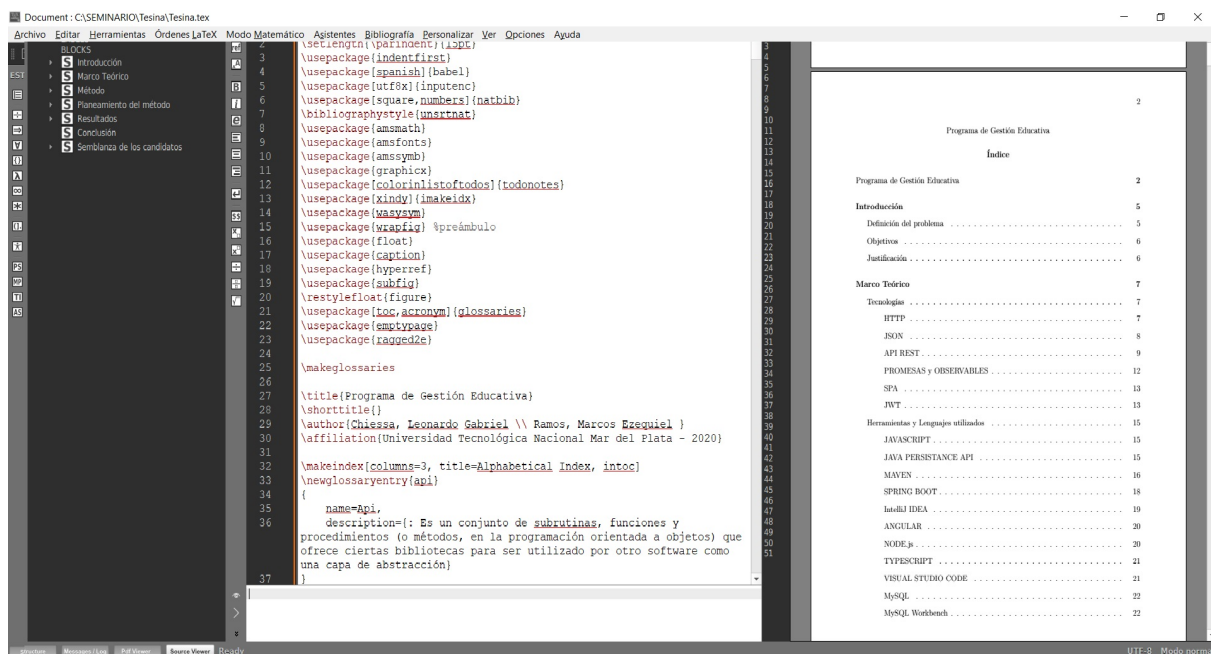


Figura 17. Gráfico #17: Entorno visual de Tesina Insufodo.

Método

Proceso de Aprendizaje

Se partió de la base de plasmar todo lo aprendido en la facultad en un solo proyecto que incluya la integración de los contenidos (Backend y Frontend) para lograr un proyecto sólido, novedoso y que cuente con escalabilidad a futuro. A eso se le sumo el desafío de usar en la aplicación una Base de datos relacional y tener abstracción de clases con la implementación de patrones de diseño[12].

Se planteó hacer algo desde cero, lo cual nos dio libertad a ambos participantes del proyecto respetando las reglas de negocio establecidas con el cliente. Desde el mes de Septiembre de 2019 se comenzó a diseñar el modelo de usuario que iba a utilizar el software. Acá estuvo involucrado el aprendizaje del manejo de Web Token por parte del equipo de desarrollo con compañeros de la facultad, los cuales aportaron sus conocimientos para la posterior resolución del problema. Esto generó motivación en el equipo.

Ese fue nuestro primer paso hacia lo que denominamos la estructura del Frontend. Desde ahí comenzamos a integrarlo con el Backend (Spring Boot Application) que habíamos armando en Julio de 2019. Durante esos meses el equipo de desarrollo se enfocó en el aprendizaje de Java y posteriormente, con esa base alcanzada, se dio comienzo al desarrollo en Angular y Node.js.

Vimos la necesidad de profundizar e investigar en algunos aspectos que se presentaron después de discutir las reglas de negocio con el cliente, algunos puntos que el equipo vio que debía resolver:

- Dos tipos de usuario (administrativo y estudiante).
- Objetos que se contienen a si mismos (materias correlativas).
- Grabar inscripciones a materias en simultaneidad de estudiantes a través de hilos de ejecución.
- Interacción con programas externos.
- Proceso de Testing.

- Deployar Landing Page.

Ademas se superó la dificultad que conlleva la Tesina, es decir toda la documentación del proyecto de software, para ello se eligió la herramienta Texmaker 5.0.2 utilizando LaTeX un sistema de composición de texto, orientado a la creación de documentos escritos que presentan una alta calidad tipográfica[11]. Por sus características y posibilidades, es usado de forma especialmente intensa en la generación de artículos y libros científicos que incluyen, entre otros elementos, expresiones matemáticas. Basándose en el modelo que propone APA 6 para la presentación y publicación de trabajos escritos en el ámbito universitario.

Por último el equipo plasmó todo lo mencionado anteriormente en Historias de Usuario, para ello se utilizó taiga.io <https://tree.taiga.io/project/leochiessa-tesis/backlog> allí se encuentra el esquema de nuestro Product Backlog.

Esta herramienta dio buenos frutos ya que rompimos un poco la estructura de algo que los programadores llamamos “programar primero, documentar después”.

Estado alcanzado

La etapa inicial de aprendizaje concluyó en tiempo y forma según el cronograma. Se logró una base sólida de conocimiento, suficiente para empezar a trabajar primero en el Backend y luego en el Frontend. Se fortalecieron conceptos claves relacionados a la programación y se mejoró la capacidad lecto-compresora del inglés.

Se produjo la integración del Backend y Frontend de manera correcta y se pudieron cumplir la mayoría de las reglas de negocio establecidas por nuestro cliente.

Al vernos limitados económicamente para deployar el backend a un hosting pago, se optó por desarrollar y hostear una Landing Page en Heroku, con el mismo diseño y estilo que la aplicación real, de manera gratuita. De esta forma el usuario estudiante envía la información de la inscripción a un email académico mediante el cual el usuario administrativo realiza la inscripción localmente en el sistema. La Tesina se fue enriqueciendo de información, conceptos, imágenes que lograron una mejor lectura e interpretación de nuestro trabajo global.

Utilizamos metodologías ágiles, específicamente SCRUM que más que una metodología de desarrollo es un esquema de trabajo. Es un conjunto de buenas prácticas que se deben aplicarse para realizar trabajos de manera colaborativa con la finalidad de tener desarrollos de calidad en el menor tiempo posible.

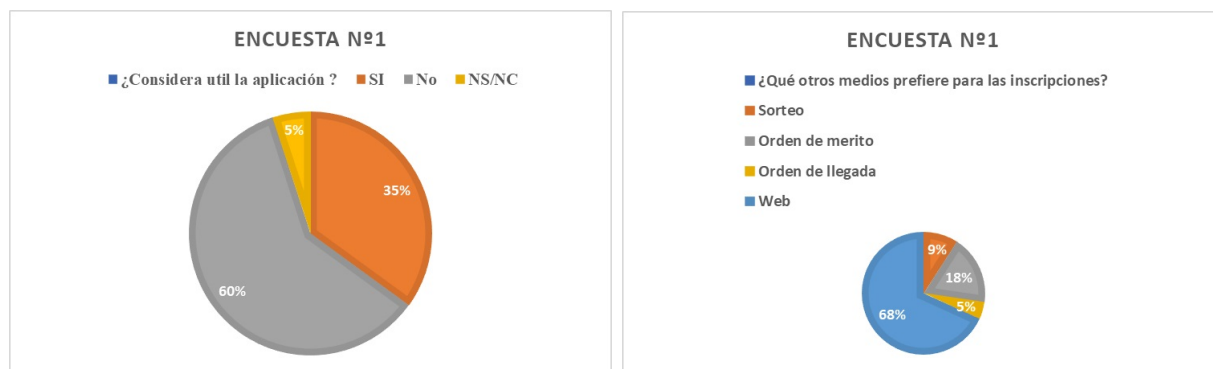
Recolección de datos mediante encuestas

Como punto de partida se realizó una encuesta a 5 profesores y 15 alumnos que utilizaron la aplicación anterior a Insufodo, instalada en el establecimiento desarrollada por la empresa marplatense denominada GrupoESI. Fue durante el cuatrimestre anterior para obtener opiniones acerca del funcionamiento y características deseadas. Se decidió que las mismas fuese anónimas para intentar obtener resultados objetivos. Contaba con 4 preguntas:

- ¿Considera útil la aplicación? ¿Qué otros medios prefiere para las inscripciones?
- ¿Cree que debe eliminarse alguna característica de la aplicación? En caso afirmativo, ¿Cuál?
- ¿Qué característica cree que debe agregarse a la aplicación en el corto plazo?
- ¿Qué característica cree que debe agregarse a la aplicación a largo plazo?

Estas preguntas fueron armadas y confeccionadas por parte del personal directivo y administrativo escolar actual con el objetivo de orientarnos y sacar sus dudas en los puntos donde el equipo de trabajo debía hacer mayor hincapié.

En la siguiente hoja de la Tesina se muestran gráficos 2D diseñados en Microsoft Excel los resultados finales de las encuestas:



(a) ¿Considera útil la aplicación?

(b) ¿Qué otros medios prefiere para las inscripciones?

Figura 18. Gráfico #18: resultados de la primera pregunta de la encuesta.



(a) ¿Cree que debe eliminarse alguna característica de la aplicación?

(b) ¿Qué otros medios prefiere para las inscripciones?

Figura 19. Gráfico #19: resultados de la segunda y tercera pregunta de la encuesta.

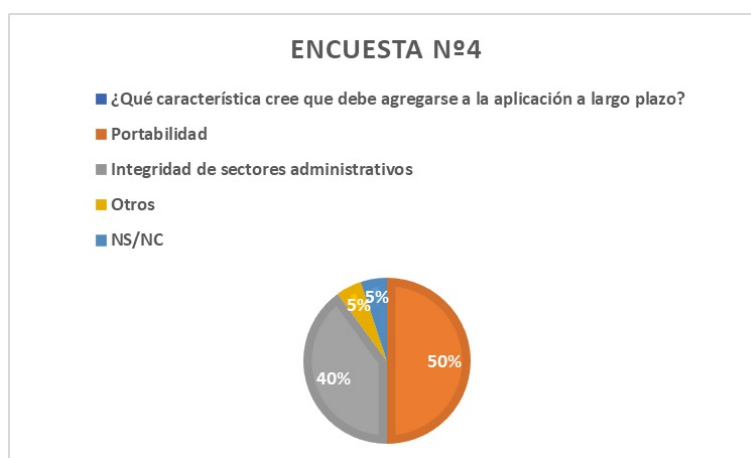


Figura 20. Gráfico #20: resultados de la cuarta pregunta de la encuesta.

Debilidades detectadas

Viabilidad

Con el fin de determinar la viabilidad del presente desarrollo utilizaremos el conocido análisis FODA (o DAFO en algunas fuentes) en el que se evalúan ventajas, desventajas, fortalezas y debilidades del tema en cuestión.

Enumeradas a continuación:

1. Fortalezas

- Existe mucho material didáctico por parte del software a usar.
- Buen conocimiento por parte de los desarrolladores en las tecnologías de software utilizadas.

2. Oportunidades

- Se ha investigado previamente y encontramos integración por parte de la problemática mencionada y su posterior resolución.

3. Debilidades

- El equipo deberá capacitarse en cuanto a las funcionalidades de diferentes usuarios (Administrativos y Estudiantes).
- Parte del equipo deberá capacitarse en cuanto a los lenguajes de programación.
- Las complejidades propuestas por el equipo directivo escolar, sobrepasan los tiempos de desarrollo previstos.

4. Amenazas

- Integración del software con las partes que lo usen (empleados administrativos de edad avanzada).

Revisión Preliminar

Antecedentes

Encontramos mucho software instalado dentro del establecimiento educativo (desde aplicaciones locales en C# hasta un prototipo realizado en PHP) ninguno atiende a la problemática planteada por el equipo directivo y brindan una solución parcial al problema mencionado al principio de la Tesina.

Sobre el lenguaje empleado en el BackEnd

Partimos de la base de Java 7 y sus similitudes con C #, para luego elevar nuestros conocimientos a Java 8 con la implementación de herramientas de última tecnología (Maven, Spring Boot, Lombok, JPA).

Sobre el lenguaje empleado en el FrontEnd

Considerando que se pretende utilizar como lenguaje Javascript (o derivados), encontramos incalculable información sobre su uso. Además, los integrantes poseen conocimiento básico.

Sobre MySQL

Ampliamente documentado en la web. Además, un integrante posee 5 años de trabajo y conocimientos.

Cronograma Product Backlog

El siguiente es un listado de los tiempos y metodologías de trabajo del equipo, formando sprints con sus tareas, asignándolas a los responsables en cuanto a su conocimiento. En esta sección recomendamos visitar el enlace de Taiga donde está detallado todo el trabajo necesario para el desarrollo y construcción de nuestro sistema. Es el resultado del trabajo del Product Owner con los diferentes interesados (cliente/s, usuario/s). Enlace : <https://tree.taiga.io/project/leochiessa-tesis/backlog>

1. Reuniones con equipo directivo escolar para el desarrollo de la aplicación web con las reglas de negocio impuestas por ellos. Se obtiene Feedback adecuado. (Duración 15 días; comienzo 16 de diciembre 2019).
2. Instanciación y configuración de Maven + SpringBoot + JPA en el que se basará el desarrollo Backend. (Duración 60 días; comienzo 2 de enero 2020).
3. Instanciación y configuración de Node.js + Angular 8 en el que se basará el desarrollo Frontend. (Duración 45 días; comienzo 2 de enero 2020).
4. Testing dedicado al funcionamiento de la aplicación (*No visibles por el usuario final*).
5. Pantallas y funcionalidad en el aplicativo Frontend para la administración (*ABM*) de la información almacenada.
6. Scripts generadores de código a partir de los datos definidos por el usuario con el fin de generar una aplicación final desarrollada.

Proceso de desarrollo

Puesta en marcha del entorno de trabajo. Se contó con dos estaciones de trabajo, una por integrante. La primera, donde fue desarrollado el prototipo inicial, una Notebook Acer Aspire provista de un procesador Intel i5-6200U, 8GB DDR3 SDRAM, disco magnético de 1TB 5400RPM y sistema operativo Windows 10. Esta computadora ya tenía instalado todo lo necesario para trabajar como servidor de la aplicación. La segunda computadora, una Pc de escritorio provista de un procesador Intel i7, 16GB DDR3 SDRAM, disco sólido de 500GB 5400 RPM y sistema operativo Windows 10. En esta computadora se instalaron no solo las aplicaciones relativas a la versión a desarrollar, sino también las necesarias para que la versión auditada pudiera ejecutarse sin inconvenientes. En detalle, se instaló IntelliJ, se instaló Visual Studio Code, se instaló MySQL Server para el manejo de base de datos, Sublime Text como editor de texto auxiliar, Node.js, Angular CLI, Heroku, Git para el control del CVS, Text Maker para el desarrollo

de la documentación en formato PDF. El resto de las bibliotecas y módulos utilizados fueron instalados a través de estas herramientas y no se listan en detalle.

Se instalaron durante el proceso, de acuerdo a disponibilidad, los navegadores Mozilla Firefox, Microsoft Edge, Internet Explorer, Google Chrome.

Spring 0. Antes de comenzar con la etapa de desarrollo, en base al listado de características a implementar y el estado de conocimiento del equipo, se tomaron decisiones que se desviaban de las planteadas durante el anteproyecto.

Como primera medida, se decidió realizar una reunión de planificación semanal los domingos a última hora, siendo este un horario cómodo que no intervenía con las responsabilidades diarias de los integrantes del equipo. A su vez y por el mismo motivo, se prefirió realizar una reunión diaria al finalizar la jornada y no al comienzo, como es recomendado en las metodologías ágiles. Aunque la mayoría de las reuniones fueron presenciales, se utilizó Google Hangouts y WhatsApp Web como medios principales de comunicación.

Se optó por continuar utilizando metodologías ágiles, pero se desistió del uso de extreme programming en favor de SCRUM. Se mantuvo la duración de cada Sprint de quince a excepción del spring numero tres que duro un mes.

Se decidió dividir el product backlog en cuatro partes correspondientes a cada sprint, en lugar de seleccionar las tareas a realizar al comienzo de cada uno. La tabla del gráfico #21 muestra la asignación tentativa de tareas:

Sprint	Tareas asignadas
Sprint 1	(3), (4), (14), (5)
Sprint 2	(6), (15), (12), (13), (7), (9), (17)
Sprint 3	(10), (11), (26), (24), (22), (25), (27), (35), (37)
Sprint 4	(46), (45), (44), (47), (48), (49)

Figura 21. Gráfico #21: Tabla asignación de tareas.

Spring 1. Se creó el repositorio en GitHub para comenzar a trabajar. La tarea (3) “análisis de modelo de negocio” se asignó a ambos integrantes. Se pactó una reunión con los directivos escolares debatiendo las reglas de negocio. La tarea (4) “elección del motor de base de datos” fue asignada a Leonardo Chiessa, como motor definitivo se optó por MySQL Server.

La tarea (14) “elección de las herramientas de desarrollo” fue asignada a Marcos Ramos, quien instaló los componentes en su PC. Se establecieron las herramientas de software a utilizar:

- IntelliJ IDEA Community Edition 2019.3.1 (Backend)
- Spring Framework 2.2.2 de Maven Central.(Backend)
- Visual Studio Code 1.41, Node.js 12.4 (Frontend)
- Angular CLI 8.3.21 (Frontend)

La tarea (5) “diseño de la base de datos” se realizó en conjunto para lo cual se creó un DER en dbdesigner.net para comenzar a orientarnos en cuanto a las entidades y relaciones.

Spring 2. La tarea (6) “creación de la base de datos” que depende de la tarea (5) se asignó a Leonardo Chiessa quien ejecutó los comandos necesarios en MySQL Workbench.

La tarea (15) “creación de usuario admin” asignada a Marcos Ramos, se hizo mediante Workbench utilizando la siguiente query: “insert into user values(‘admin@insufodo’, ‘asd’)”.

La tarea (13) ”login” fue asignada a Leonardo Chiessa, quien se encargó de su desarrollo. En el backend se creó modelo, repositorio y controladora, así como también el End Point de Web Security donde se encuentra la lógica para generar el Token; y en el frontend se creó modelo, servicio y componente de Angular.

La tarea (12) “alta de usuario administrativo” fue asignada a Marcos Ramos. Al tener el esquema armado en la tarea anterior, se clonó la estructura que permitió darle vida a este componente tanto en el backend como en el frontend.

La tarea (7) “abm de alumno” fue realizada por Marcos Ramos basándose también en las tareas (13) y (12). Lo mismo ocurrió con la tarea (9) “abm de materia” que fue

realizada por Leonardo Chiessa.

A medida que avanzábamos en el desarrollo, fuimos descubriendo que eran necesarias ciertas validaciones del lado del cliente, ya que nos lo exigían las reglas de negocio. Para resolver este tema se usaron algunos Async Validators.

Spring 3. Las tareas (10) “alta cohorte” y (24) “alta final” fueron asignadas a ambos miembros ya que era necesario, aparte de la programación, aplicar cierto conocimiento en cuanto a la problemática escolar y la lógica de correlatividades del plan de estudios.

Una vez hecho el CRUD (create, read, update y delete) de cohortes, se procedió con la tarea (11) “inscribir alumno” que fue realizada por Leonardo Chiessa también creando modelo, repositorio, controladora, servicio y componente.

La tarea (26) “carga de calificaciones” fue asignada a Marcos Ramos, quien a partir de la tarea (10) se encargó del update de los campos `qualification` y `qualification_date` de la tabla `inscription`.

Tarea (22) “inscripción a final” ídem (11) y tarea (25) “asignación de notas de final” ídem (26). La tarea (37) “listar inscriptos por cohorte” fue hecha por ambos miembros ya que llevó bastante tiempo y complejidad y terminó dando como resultado la vista del Estado Académico del Alumno.

La tarea (35) “tesina” pasó para el cuarto sprint ya que la dificultad y las modificaciones constantes, sobrepasaron los tiempos estipulados. Esta tarea fue realizada en conjunto por ambos miembros durante todo el proceso.

La tarea (27) “recibida de alumno” por el momento queda archivada, ya que no se encuentra dentro de los objetivos planteados.

Spring 4. Tarea (44) “hostear landing page”. Esta historia surgió luego de una primera reunión con el docente German Gianotti de la materia Seminario, quien nos recomendó esta solución al problema de costos. Fue realizada y testeada por ambos miembros mediante la herramienta Heroku.

La tarea (45) “crear funcionabilidad para manual de usuario” fue asignada a Leonardo Chiessa quien creó un manual de usuario administrativo en formato pdf para descargar desde dentro de la aplicación y un manual de estudiante en formato mp4 para descargar

desde el login orientando de esta forma a ambos usuarios.

La tarea (46) " test unitarios backend " fue asignada a Marcos Ramos. Se obtuvo un 90 % de cobertura en la controladora.

La tarea (47) "crear servicios del lado del backend" fue asignada a Leonardo Chiessa quien ordeno los métodos con su correspondiente lógica.

La tarea (48)" paginar listados" fue asignada a Leonardo Chiessa se implementó JPA pagination en el backend y en el frontend se utilizó un componente tipo AppPaginator.

La tarea (49)"finalizar y exportar productbacklog para agregar en la tesina" fue asignada a Marcos Ramos. Desde taiga.io se exporto el Product Backlog en formato Json.

La tarea (50) "enviar mail de notificación de calificación a alumno" fue realizada por ambos miembros. Desde el backend se implementó JavaMailSender que envía un email al estudiante cuando es calificado.

Planeamiento del método

En una primera etapa, partiendo de la disparidad de conocimientos del equipo de desarrollo, nos proponemos nivelar el entendimiento y la aplicación de las herramientas y tecnologías utilizadas. Este es un proceso ya comenzado que nos llevo un mes de trabajo y comunicación. En paralelo, realizaremos una encuesta al personal docente encargado de la administración de alumnos de los Institutos de Formación Docente de Nivel Superior de la ciudad, a quienes les gustaría tener un sistema de esta índole. Luego, procederemos al análisis de los datos recaudados, enfocados en nuestro sistema, buscando cambios y pensando en patrones, código, base de datos, pruebas y documentación.

Concluido el proceso de estudio y análisis, nos proponemos realizar el desarrollo del software aplicando metodologías ágiles. Además, con el ideal de codificar primero la prueba de algún módulo que creamos esencial, comenzando por el backend en IntelliJ para luego dar paso al frontend en Angular y VisualStudio Code. Esto nos permitirá alcanzar el mínimo de cobertura propuesto para las pruebas unitarias incluidas.

En materia de comunicación utilizaremos Taiga, WhatsApp Web y Hangouts aunque, ante el creciente abanico de posibilidades, estaremos probando otras opciones durante el desarrollo. Para realizar un control de versiones sobre todo nuestro trabajo, utilizaremos GitHub. Desarrolaremos nuestro trabajo en 2 partes: Backend (Spring Boot, gestor de proyectos Maven, JPA) y Frontend (Desarrollo en Angular y Typescript). Servidor de Base de Datos: MySQL. Dividiremos todo en Sprints, asignándonos a cada uno distintas tareas, pero colaborando ambos con todas.

Composición general tentativa o guion

A continuación presentamos un índice tentativo de los temas a cubrir en el documento final de la Tesina para el presente proyecto:

1. Introducción.
 - a) Planteamiento del Problema.
 - b) Justificación.

c) Objetivos.

d) Viabilidad del desarrollo.

2. Planeamiento del método.

a) Definición de requerimientos.

b) Desglose de tareas y board interactivo.

c) Estimación y diagramas.

d) Sobre la metodología de desarrollo.

e) Sobre los estándares de codificación

3. Sobre el lenguaje elegido para el desarrollo.

a) Reseña

b) Fundamentos

c) Características

4. Sobre las herramientas utilizadas para el desarrollo.

a) Versionado.

1) GIT.

2) Github.

b) Codificación.

1) API Rest.

2) Angular(Typescript)

c) Organización de tareas y seguimiento.

1) Taiga.

5. Documentación de instalación y uso de la aplicación.

a) Instalación.

- b)* Configuración de un entorno físico.
 - c)* Configuración de un entorno de desarrollo.
 - d)* Generando una api básica.
 - e)* Validaciones de datos.
 - f)* Desarrollo.
6. Conclusión final.

Resultados

Documentación

Se generó documentación para los diferentes consumidores del proyecto. En primera medida, se entrega vídeos y documentación alojados en el servidor del proyecto. Explicativos de usuario detallando el funcionamiento completo del sistema y sus diferentes módulos, así como sus restricciones y formas de utilizarlos. Estos vídeos constan de dos apartados, uno para los administrativos y otro para los alumnos.

Por otro lado, a partir de los comentarios en el código fuente, se generó una referencia completa de la API. A pesar de que no sea pública, merece su extracción para facilitar la lectura y comprensión del código por parte de revisores y futuros colaboradores.

Implementación

Se utilizaron las últimas tecnologías disponibles y buenas prácticas de programación. Se estructuró y modularizó el código, se aplicó todo el conocimiento aprendido durante el proceso. Se trabajó con Microsoft OneDrive como CVS y GitHub como repositorio central <https://github.com/leochiessa/Insufodo> , por lo que el código fuente se encuentra disponible en línea para cualquiera que desee utilizarlo.

Pruebas

Se trabajó fuertemente en incorporar al proceso de desarrollo las pruebas de software, tanto unitarias como funcionales. Se analizó la utilización de pruebas automáticas, pero se decidió posponer su implementación para las próximas etapas de crecimiento.

Con respecto a las pruebas unitarias, se logró una cobertura superior al 85 %, superando el objetivo del 70 % propuesto durante las etapas iniciales. Se decidió trabajar en las próximas etapas para alcanzar el grado de cobertura superior al 95 %.

Rendimiento

Finalizadas las tareas propuestas se procedió a analizar el rendimiento de la versión final, dichas tareas fueron testeadas en su rendimiento en los tiempos de carga. Se recomienda una buena conexión a Internet disminuyendo los tiempos de peticiones a End-points. También se analizó el consumo de memoria Ram y procesador del servidor para tareas como el logueo, donde se recomienda un ordenador de características de uso para el desarrollo (Pentium I5 en adelante, 8Gb de Ram recomendado) para que la herramienta funcione de manera ágil. Estos buenos resultados son consecuencia de las nuevas tecnologías elegidas para el desarrollo del mismo haciendo una aplicación fuerte y robusta al alcance de cualquier establecimiento educativo de la comunidad.

Conclusión

Se finalizó el desarrollo en tiempo y forma cumpliendo los objetivos propuestos, descubriendo nuevas tecnologías, profundizando habilidades y conocimientos, mejorando capacidades personales y codificando finalmente una aplicación funcional de calidad acorde a lo planteado.

Se sentó una sólida base de trabajo. El análisis realizado, las falencias detectadas y las debilidades reforzadas, brindan un excelente punto de partida para continuar el proceso de desarrollo. Las encuestas realizadas, así como las investigaciones relativas a las diferentes tecnologías entregaron un amplio abanico de posibilidades para la búsqueda de características a implementar en el futuro.

Se debatió acerca de diferentes aspectos relativos a las tecnologías utilizadas, se revisó y reutilizó lógica y código ajeno anterior a Insufodo, se tradujo documentación oficial, lo cual abrió nuevas puertas para el desarrollo personal.

A partir de este punto, se implementará el sistema desarrollado en un ambiente real, donde pueda probarse a fondo, explotar sus fortalezas y resaltar sus debilidades. Se trabajará en conjunto con alumnos y profesores para mantener un ciclo de mejoras continuas, añadiendo funcionalidades deseadas. Se mantendrá el espíritu libre y superador del proyecto, abierto a contribuciones externas.

Se debe recordar, sin embargo, que la tecnología es solo un medio hacia un fin. Son los empleados administrativos y profesores quienes deben guiar a los alumnos apoyándose en esta herramientas y no dependiendo de ella.

Semblanza de los candidatos

Chiessa, Leonardo Gabriel

Argentino, 33 años, estudiante de la carrera de **Técnico Universitario en Sistemas Informáticos** de la **UTN Argentina**. Actualmente dedicado profesionalmente al desarrollo de software desde los 29 años de edad hasta la fecha en un hospital privado.

Ramos, Marcos Ezequiel

Argentino, 33 años, estudiante de la carrera de **Técnico Universitario en Sistemas Informáticos** de la **UTN Argentina**. Actualmente dedicado profesionalmente a la docencia y funciones administrativas escolares. Desde los 27 años de edad hasta la fecha desarrolla su actividad en el Instituto de Formación Docente N°84 de la ciudad de Mar del Plata.

Siglas

API Application Programming Interface. 9

CSS Cascade Style Sheet. 12

DOM Document Object Model. 11

FODA Fortalezas Oportunidades Debilidades Amenazas. 30

HTML HyperText Markup Language. 9

HTTP Hypertext Transfer Protocol. 12

IDE Integrated Drive Electronics. 18

JAR Java ARchive. 16

JPA Java Persistence API. 14

JPQL Java Persistence Query Language. 15

JSON JavaScript Object Notation. 8

JWT JSON Web Token. 12

POM Project Object Model. 15, 16

REST Representational State Transfer. 9

SPA Single Page Application. 12

URI Uniform Resource Identifier. 6

WAR Web Application Archive. 16, 17

XML eXtensible Markup Language. 15

Glosario

Applets de Java : Representan un método versátil para incluir contenido multimedia en las paginas Web.Interacción con el usuario . 10

Blink : Motor de renderizado desarrollado por Google. Toma contenido marcado (como HTML, XML, archivos de imágenes, etc.) e información de formateo (como CSS, XSL, etc.) y luego muestra el contenido ya formateado en la pantalla de aplicaciones. Es utilizado por varios navegadores: Chromium, Chrome, Opera, Brave, Vivaldi, Maxthon y Microsoft Edge . 20

Dom : Interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML y XML,un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente. 11

Electron : Permite el desarrollo de aplicaciones gráficas de escritorio usando componentes del lado del cliente y del servidor originalmente desarrolladas para aplicaciones web: Node.js del lado del servidor y Chromium como interfaz. Electron es el framework gráfico detrás de muchos proyectos de código abierto importantes. 20

Framework : Conjunto de herramientas y utilidades basadas que sirve como base para la organización y el desarrollo de software. 14

Html : Lenguaje de marcado utilizado principalmente en la elaboración de páginas web. 14

Http : El Protocolo de transferencia de hipertexto, es el protocolo de comunicación que permite las transferencias de información en la World Wide Web. 7

IETF : Comunidad internacional abierta de diseñadores de redes, operadores, vendedores e investigadores preocupados por la evolución de la arquitectura de Internet y el buen funcionamiento de Internet. 6

IntelliSense : término general para una variedad de características de edición de código que incluyen: finalización de código, información de parámetros, información rápida y listas de miembros. Las funciones de IntelliSense a veces se llaman por otros nombres como "finalización de código", "asistencia de contenido" y "sugerencia de código". 21

Landing Page : Página web preliminar o pagina de aterrizaje donde se quiere destacar algo en especial, ya sea un producto, o alguna novedad o promoción de un producto. 25

LaTeX : Sistema de composición de textos orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Por sus características y posibilidades es usado de forma especialmente intensa en la generación de artículos y libros científicos. 27

MIME : Serie de especificaciones dirigidas al intercambio a través de Internet de todo tipo de archivos de forma transparente para el usuario. 6

Modelo OSI : Interconexión de Sistemas Abiertos. Es un modelo o referente creado por la Organización Internacional de Normalización para la interconexión en un contexto de sistemas abiertos. Se trata de un modelo de comunicaciones estándar entre los diferentes terminales y host. 6

Script : Archivo de órdenes, archivo de procesamiento por lotes, es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. Los guiones son casi siempre interpretados, pero no todo programa interpretado es considerado un guion. El uso habitual de los guiones es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario. 10

URI : Cadena de caracteres que identifica los recursos de una red de forma unívoca. 10

WWW : Sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles a través de internet. 6, 9

Referencias

- [1] R. Fielding, J. Gettys, J. Mogul, J. Mogul, L. Masinter, P. Leach, T. Berners-Lee. *"Hypertext Transfer Protocol – HTTP/1.1"*. Recuperado : <https://www.w3.org/Protocols/rfc2616/rfc2616.html>. RFC 2616. The Internet Society 1999.
- [2] T. Bray, Ed. *"The JavaScript Object Notation (JSON) Data Interchange Format"*. Recuperado : <https://tools.ietf.org/html/rfc8259>. RFC 8259. 2015.
- [3] Fielding, Roy Thomas. *'Estilos arquitectónicos y el diseño de arquitecturas de software basadas en red'*. Recuperado : <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. Tesis doctoral, Universidad de California, Irvine. 2000.
- [4] V.Jones M., Bradley J., Sakimura N. *"JWT Json Web Token"*. Recuperado : <https://tools.ietf.org/html/rfc7519>. RFC 7519. 2015.
- [5] Flanagan, David. *"JavaScript - The Definitive Guide", 5th ed.*. O'Reilly, Sebastopol, CA, 2006.
- [6] Cecilio Alvarez Caules. *'Arquitectura Java JPA Domain Driven Design'*. Caules, C.A. 2014.
- [7] Company, S. *"Maven: The Definitive Guide"*. O'Reilly Media. 2008.
- [8] Ryan Dahl. *'Introduction to Node.js with Ryan Dahl'*
https://www.youtube.com/watch?v=jo_B4LTHi3I
- [9] Documentación oficial TypeScript,
<https://www.typescriptlang.org/docs/home.html>
- [10] Documentación oficial Visual Studio Code,
<https://code.visualstudio.com/docs>
- [11] Michel Goossens, Frank Mittelbach, and Alexander Samarin. , *"The L^AT_EX Companion"*. Addison-Wesley, Reading, Massachusetts, 1993.

- [12] Freeman, Eric and Robson, Elisabeth and Bates, Bert and Sierra, Kathy. , *"Head First Design Patterns: A Brain-Friendly Guide"*. O'Reilly Media, Inc. 2004.
- [13] Documentación oficial Spring Boot,
<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
- [14] Documentación oficial Angular,
<https://angular.io/docs>
- [15] Documentación oficial MySQL,
<https://dev.mysql.com/doc/>