UNIVERSIDAD DEL BÍO-BÍO

FACULTAD DE CIENCIAS EMPRESARIALES

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN Y TECNOLOGÍAS

DE LA INFORMACIÓN

Desarrollo de servicios REST utilizando el Framework Spring Boot: Un material didáctico para la asignatura de Ingeniería de Software.

Trabajo para optar al título de Ingeniero Civil en Informática.

Mauricio Andrés Sepúlveda Méndez

Docente Guía: Rodrigo Ariel Torres Avilés

Agradecimientos

A mis viejos

Por estar siempre para darme ese apoyo incondicional y mostrarme de que sin importar nuestras diferencias, iban a estar allí, destacando a mi Vieja con su paciencia infinita y a mi Viejo por darme el ejemplo de que con esfuerzo se pueden lograr muchas cosas.

A mis hermanos

A Alejandro por enseñarme como desarmar un computador y que al armarlo no prenda.
A Kristian, por ser él, realmente no hizo nada, pero quería ponerlo igual.

A mis amigos y compañeros

A todos con los que he compartido una risa, una cerveza, un buen churrasco o una pizza, Gonzalo, Cristóbal, Hilda, Esteban, Camilo, Carlos, Luis, Marietta, Catalina, España, Kattya, Claudio, Carla, Cristian, Alan, Claudia, Ramiro, Jessica, Cristian, Macarena, Sebastián, Diego, Remigio, Ignacio, Estefanía, Cruzana, Gustavo, Felipe, Vinka, Bárbara, Carlos, Pablo, y tantos más que la lista se hace interminable, gracias a todos por apoyarme en esta aventura.

A los que cayeron

Ya que con ellos reprobar siempre fue más divertido, Leonardo, Felipe, esta aventura fue mucho más entretenida gracias a ustedes.

A mis colegas, que también son mis amigos

Que cada día comparten conmigo de sus conocimientos y experiencias, permitiendo que pueda mejorar como profesional, gracias, Claudio, Germán, Laura, Raymi, Henrique, María-Jesús, Felipe, Camila, Piero, Sayaka, Javier, Henry, Alexander, Lino, Carlos y muchos más, gracias a todos.

A mis profesores

Que estuvieron allí cada día de mi formación profesional, apoyándome, aconsejándome y dándome las herramientas que hoy en día me permiten hacer lo que me apasiona.

Y a ti, por leer hasta los agradecimientos hasta el final.

Resumen

Este proyecto se presenta para dar conformidad a los requisitos exigidos por la Universidad del Bío Bío, en el proceso de titulación de la carrera de Ingeniería Civil en Informática.

El proyecto titulado "Desarrollo de servicios REST utilizando el Framework Spring Boot: Un material didáctico para la asignatura de Ingeniería de Software" tiene como objetivo presentar un material de apoyo para el estudiante durante el desarrollo del "portafolio del proyecto de desarrollo de software", que contempla el programa de la asignatura de Ingeniería de Software de la carrera Ingeniería Civil en Informática.

Lo previamente expuesto, se llevará a cabo mediante la entrega de conocimientos y el uso de diferentes metodologías, técnicas y tecnologías, las cuales han sido altamente demandadas en la industria del software actual. Además, se entrega un conjunto de actividades recomendadas con las cuales el docente podrá instar a los estudiantes a desarrollar de mejor manera los contenidos expuestos dentro del aula.

Para la gestión de este proyecto, se optó por utilizar una adaptación de la metodología tradicional de desarrollo Iterativa Incremental, en la cual se añadieron diversos elementos de la metodología ágil Scrum, y en lo que concierne para el desarrollo de los ejercicios presentados dentro del material didáctico se utilizaron técnicas como el Test-Driven Development.

Con el desarrollo de este proyecto se logró construir un material de estudio el cual comprende diferentes aspectos del desarrollo de proyectos de software, desde las metodologías de desarrollo ágiles, la gestión de proyectos, técnicas para el desarrollo de software tanto individuales como grupales y además cubre tecnologías que en la industria de hoy en día están siendo requeridas.

Abstract

This project is presented to comply with the requirements of the Universidad del Bío Bío, in the process of qualification of the Civil Engineering degree in Computer Science.

The project entitled "Desarrollo de servicios REST utilizando el Framework Spring Boot: Un material didáctico para la asignatura de Ingeniería de Software" aims to present a support material for the student during the development of the " portafolio del proyecto de desarrollo de software", that contemplates the program of the Software Engineering subject of the Civil Engineering in Computer Science career.

The foregoing will be carried out through the delivery of knowledge and the use of different methodologies, techniques and technologies, which have been highly demanded in the current software industry. In addition, a set of recommended activities is delivered with which the teacher can encourage students to better develop the contents exposed in the classroom.

For the management of this project, we chose to use an adaptation of the traditional method of incremental Iterative development, in which various elements of the Scrum agile methodology were added, and in what concerns the development of the exercises presented within the material didactic techniques such as Test-Driven Development were used.

With the development of this project it was possible to build a study material which includes different aspects of the development of software projects, from the agile development methodologies, project management, techniques for the development of individual and group software and also covers technologies that in today's industry are being required.

Índice General

Índice Tablas

Índice Figuras

1 INTRODUCCIÓN

La velocidad en la cual la industria del software evoluciona hace que hoy en día sea muy difícil llevarle el ritmo, ya que en los últimos años han aparecido nuevos conceptos, técnicas, y metodologías para trabajar en este rubro, lo que ha generado una cantidad importante de cambios en comparación a como se trabajaba hace algunos años atrás.

A lo largo del presente trabajo, se presenta la estructura del desarrollo de un material de didáctico para su uso durante la asignatura "Ingeniería de Software", con el cual se busca apoyar tanto a docentes como a estudiantes entregándoles material de estudio utilizando metodologías, tecnologías y técnicas, que hoy en día son demandadas por la industria del software[1].

Es por esto que este trabajo tiene como objetivo potenciar las habilidades de los estudiantes en el ámbito del desarrollo de proyectos de software cubriendo distintas áreas de este entorno, considerando desde la metodología a utilizar, la forma de expresar los requerimientos del cliente, como distribuir de manera ordenada las versiones de un proyecto, además de formas de potenciar el trabajo en equipo y finalmente presentando herramientas que hoy en día están teniendo un alto impacto en la industria del desarrollo.

El trabajo se estructura como sigue: En el segundo capítulo se darán a conocer los detalles de la institución educativa para la cual se desarrollará una solución al problema que exponen los estudiantes y una solución a alto nivel de este. Durante el tercer capítulo se hablará sobre la definición del problema, y el ambiente ingenieril con el que este será enfrentado. El capítulo cuarto expondrá detalles generales sobre el producto presentado como las razones de la elección del contenido, la estructura de cada uno de los capítulos del material de estudio y sus contenidos a un nivel bastante general. Después, en el quinto capítulo se presentarán los detalles sobre el material de estudio, un resumen de los objetivos específicos de cada unidad, la forma en la cual se detallaron los requerimientos obtenidos. El sexto capítulo describirá los distintos tipos de ejercicios en los cuales era necesario desarrollar algún tipo de Software. Luego, en el séptimo capítulo se presenta la opinión de profesionales de la industria respecto al material de estudio. Finalmente, en el octavo y último capítulo, se entregan las conclusiones y una proyección del trabajo a futuro.

[1] Según busquedas individuales de cada uno de los temas tratados dentro de este trabajo en plataformas como GetOnBrd

2 DEFINICIÓN DE LA EMPRESA O INSTITUCIÓN

2.1 Descripción de la institución:

El Departamento de Ciencias de la Computación y Tecnologías de la Información (DCCTI), el cual pertenece a la Facultad de Ciencias Empresariales (FACE) de la Universidad Del Bío Bío del Campus Fernando May el cual está ubicado en la ciudad de Chillán, tiene los siguientes antecedentes:

Antecedentes de la Institución:

- Nombre: Universidad del Bío Bío.
- Nombre Departamento: Departamento de Ciencias de la Computación y Tecnologías de la Información.
- Dirección: Avenida Andrés Bello #720, Chillán.
- Rubro: Educación.
- Servicios que ofrece: Servicios de Educación Superior.
- Página web: http://www.ubiobio.cl/dccti/

Competencia directa:

Institutos de educación superior que imparten carreras relacionadas a la industria de la informática dentro de la ciudad de Chillán, tales como:

- Inacap.
- Santo Tomas.
- Virginio Gómez.
- Universidad Adventista de Chile.

2.1.1 Visión

"Ser reconocido por su aporte científico-académico en el área de Ciencias de la Computación y Tecnologías de Información a nivel nacional, con un cuerpo académico de excelencia que conforme un grupo de investigación consolidado, integrado a redes científicas, académicas, y adecuadamente vinculado con el medio". (DCCTI)

2.1.2 Misión

"Cultivar las disciplinas de Ciencias de la Computación y Tecnologías de Información, a fin de contribuir en la formación de profesionales a través de loa docencia de pre y postgrado, y difundir el conocimiento asociado a estas disciplinas, tanto a la comunidad científica como a la sociedad en su totalidad, aportando, de este modo, al quehacer de la Universidad del Bío Bío". (DCCTI)

2.1.3 Descripción de las funciones

Preparar profesionales capacitados para la industria, mediante la capacitación constante del personal académico del departamento, también así mismo la búsqueda de acreditaciones que certifiquen la calidad de la institución de forma que se pueda marcar la diferencia con respecto a las otras instituciones de educación superior.

2.2 Descripción de la problemática

Los estudiantes de la carrera de Ingeniería Civil en Informática desde el año 2016 en adelante presentaron diferentes dificultades respecto a la asignatura de Ingeniería de Software, información la cual se comprobó mediante una encuesta anónima[2] realizada durante el segundo semestre del año 2017 y el primer semestre del año 2018.

Entre los resultados entregados[3] por los estudiantes, cerca del 46% de los alumnos encuestados presentaron problemas importantes para poder completar el proyecto de la asignatura y solo el 7,7% se sintió a gusto con su desempeño en el proyecto.

Los estudiantes encuestados expresaron mayoritariamente las siguientes razones para referirse a su desempeño respecto al proyecto del curso:

1. El poco tiempo de parte del personal docente para explicar de forma clara el cómo utilizar las nuevas herramientas que fueron propuestas para el desarrollo del proyecto.
2. La poca relación entre el material presentado dentro del aula y lo que los estudiantes necesitaban para poder desarrollar el proyecto.
3. La falta de motivación de parte de los equipos por problemas internos.

Además de estas afirmaciones, se les consulto a los estudiantes respecto a si creen que su experiencia en la asignatura hubiese sido mejor si se cuenta con un material de estudio adicional, en donde se pueda consultar respecto a los contenidos básicos para apoyar el desarrollo del

[2] Revisar anexo 1, Encuesta.
[3] Revisar anexo 2, Análisis resultados.

proyecto de la asignatura, y los estudiantes consultados tuvieron cerca de un 88% de aprobación respecto a dicha propuesta.

2.3 Solución general de la problemática

Se le ofrece a la Escuela de Ingeniería Civil en Informática un nuevo material de estudio para la asignatura de Ingeniería de Software impartida en la carrera de Ingeniería Civil en Informática, en donde se entrega un material de trabajo para los estudiantes respecto a metodologías ágiles, tecnologías y técnicas de desarrollo de Software. El material deberá estar constituido por cuatro partes para cada una de sus unidades las cuales corresponden a una sección de lectura, actividades, autoevaluación y referencias del contenido, puntualmente el material didáctico comprende los siguientes contenidos:

1. Introducción a la metodología de desarrollo Scrum.
2. Toma de requerimientos y estimación de historias de usuario.
3. Introducción a los Sistemas de Control de Versiones.
4. Code Review y Test-Driven Development.
5. Spring Boot y Servicios Rest.
6. Desarrollo de un Servicio Web del tipo REST con Spring Boot.

Dichas tópicos fueron escogidos según la popularidad de su uso y el impacto que han tenido en los últimos años en la industria. Scrum fue seleccionado por el fuerte impacto que ha tenido en las metodologías ágiles desde sus comienzos (Schwaber & Sutherland, 2017). Los sistemas de control de versiones son de vital importancia, como lo deja claro GitHub que posee más de 74 millones de proyectos alojados en sus servidores (GitHub). Para apoyar el proceso de desarrollo del Software, se seleccionaron dos técnicas que se creen fundamentales para fomentar el trabajo en equipo las cuales son el Code Review y el Test-Driven Development, ya que estas permitirán al equipo de trabajo compartir los conocimientos (Fowler, 1999) y además desarrollar el proyecto manteniendo un grado de confianza entre sus compañeros de equipo gracias a los test unitarios desarrollados (Jurado, 2010). El Framework Spring Boot ha tenido un gran crecimiento en el último tiempo pasando a ser una de las principales opciones para el desarrollo de aplicaciones de nivel empresarial con Java en los últimos tiempos (Ryan, 2017).

Además esta selección de contenido se apoya en el Descriptor de Competencias expuesto en el esquema extenso de la asignatura de Ingeniería de Software[4] en la sección II.2, la cual contempla las diferentes metas de la asignatura, en la cual se especifica explícitamente la necesidad de implementar un Software utilizando diferentes, técnicas, herramientas y métodos, de forma de construir un producto de software capaz de cumplir los requerimientos funcionales y no-funcionales del cliente.

Así mismo, en la sección II.3 del mismo documento, se consideran los aprendizajes previos para el correcto desempeño de los estudiantes en la asignatura, lo que implica que estos al cursar Ingeniería de Software deberán tener conocimientos sobre el modelado de datos, creación de software con diferentes lenguajes de programación, manejo de bases de datos y además el deben ser capaces de reconocer el contexto del negocio en el cuál se insertará la aplicación.

Adicionalmente a los criterios considerados en el plan curricular de la asignatura se considera que los estudiantes que están cursando Ingeniería de Software tienen un manejo considerable del idioma extranjero inglés ya que es una asignatura que pertenece al segundo semestre del cuarto año de la carrera, por lo que los estudiantes deben ser capaces de comprender textos en dicho idioma.

[4] Revisar Anexo 3, Esquema Extenso – Ingeniería de Software (2013)

3 DEFINICIÓN PROYECTO

En la siguiente sección se detallarán los diferentes aspectos del presente proyecto, en donde se dará a conocer sus objetivos, ambiente de desarrollo, la especificación de las tecnologías, metodologías y prácticas utilizadas durante el desarrollo del producto, además de entregar el detalle de las diferentes abreviaciones y/o siglas utilizadas.

3.1 Objetivos del proyecto

El presente proyecto consiste en la elaboración de un material didáctico para su uso dentro de la asignatura de Ingeniería de Software, en el cual se cubre el proceso de desarrollo de una aplicación que provea servicios web del tipo REST (*web services*), utilizando: la metodología de Scrum, el Framework Spring Boot y la aplicación de técnicas como Code Review y *Test-Driven Development* durante el desarrollo del proyecto.

Objetivos específicos:

- Generar material teórico-práctico para la construcción de un sistema mediante el uso de la metodología de desarrollo Scrum, y el Framework Spring Boot, el cual este orientado al uso de estudiantes de la asignatura Ingeniería de Software.
- Presentar herramientas para el control de versiones como una solución al problema de compartir y realizar seguimiento del código fuente.
- Desarrollar un Prototipo de Servicio web utilizando la técnica para desarrollo, Test-Driven Development y Spring Boot.

3.2 Ambiente del Desarrollo del Proyecto

Durante el desarrollo del material de estudio se utilizó una adaptación de una metodología iterativa incremental, en la cual se añadieron ciertos aspectos de las metodologías agiles, como la representación las tareas mediante un tablero ágil[5], con el objetivo de construir un material didáctico conforme al modelo educativo de la Universidad del Bío-Bío y a la educación basada en competencias.

[5] https://trello.com/

3.2.1 Metodología para la construcción del material de estudio

Para el desarrollo de este proyecto se utilizó gran parte de lo propuesto por la metodología de desarrollo en cascada a la cual se le añadieron diferentes toques de metodologías ágiles, como la administración de las tareas por historias de usuario, un tablero ágil, reuniones seguidas con los *stackeholders*[6] y el cliente.

3.2.2 Modelo Educativo de la Universidad

En el texto del modelo educativo de la Universidad del Bío-Bío, se presenta en la descripción de este un párrafo que le da realce al material de estudio presentado en este proyecto, ya que este documento declara la intención de la universidad de generar profesionales íntegros y de excelencia a la sociedad, como se puede leer del siguiente párrafo:

"El Modelo Educativo de la Universidad del Bío-Bío centra su acción en el estudiante, asumiendo su realidad y sus potencialidades, a partir de las cuales proyecta su formación profesional y personal, para alcanzar un desarrollo integral que le permita insertarse en la sociedad como un profesional de excelencia, dando respuesta, así, a las necesidades regionales, nacionales e internacionales". (Vicerrectoría Académica., 2008)

Razón por la cual el material de estudio presentado busca entregar las herramientas necesarias a los estudiantes para que estos puedan tener mejor rendimiento dentro de la asignatura aludida y además de que estos puedan obtener experiencias utilizando lo que se requiere hoy por hoy en la industria.

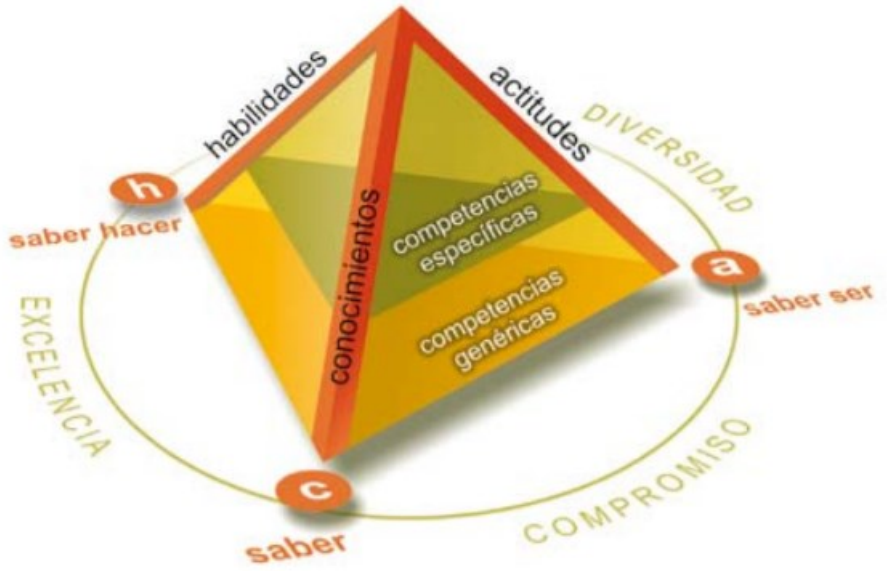
3.2.3 Educación basada en competencias

El modelo educativo de la Universidad propone además seguir un modelo basado en las competencias para la formación del estudiante, en donde las competencias son un conjunto de habilidades, conocimientos y ámbitos tanto profesionales como sociales.

"La formación por competencias en el marco del Proceso de Enseñanza y Aprendizaje asume un enfoque integral, que vincula la educación con el mundo laboral y eleva el potencial de los sujetos, de cara a las transformaciones que sufre el mundo actual y la sociedad contemporánea.

Los componentes que constituyen la Pirámide de formación por competencias son los conocimientos, habilidades y actitudes, las que se articulan en un proceso permanente de aprendizaje y desarrollo integral de competencia, que viene a constituir la combinación de saberes en la acción: saber, saber hacer y saber ser".(Ver Figuras 1.)

[6] Personas de interés en el proyecto, en este caso: estudiantes, profesionales de la industria TI, etc.



Figuras 1: Pirámide de Competencias del Egresado UBB[7]

Las dimensiones de la aplicación de las competencias representadas en la Figuras 1, consideran lo siguiente:

- Conocimiento: que representa el saber comprender, analizar y tratar la información relevante, relacionar o sintetizar está para el estudio de un determinado sistema o fenómeno.
- Habilidades: son las competencias que permiten al estudiante tomar decisiones de cómo proceder en el entorno en el que se encuentra, ya sea manejando la información, seleccionando tecnologías, etc.
- Actitudes: es parte de las competencias que le permitirá al estudiante cooperar con otros en función de un objetivo en común.

Esto según es presentado en la documentación del Modelo Educativo de la Universidad (Vicerrectoría Académica., 2008).

[7] Obtenida documento del Modelo Educativo de la Universidad del Bío-Bío.

3.2.4 Rubricas de evaluación

Durante el desarrollo de este trabajo, también se sugiere el uso de rubricas del tipo analíticas, ya que este tipo de rubricas, desglosan los componentes a evaluar para generar una evaluación final (Gatica-Lara & Uribarren-Berrueta, 2013), de este modo los estudiantes podrán tener evidencias claras de su progreso a lo largo del curso.

Este tipo de rubrica presenta tres características importantes:

1. Criterios de Evaluación: Son los factores que determinan la calidad de trabajo del estudiante. También conocidos como indicadores o guías. Estos reflejan los contenidos que se juzgan de importancia.
2. Definiciones de calidad: Son una serie de criterios que sirven para calificar el nivel de eficiencia del estudiante respecto a un tópico determinado, además, estos criterios deben servir para poder proporcionar una retroalimentación adecuada al estudiante.
3. Estrategias de puntuación: Se consideran cuatro niveles, un desempeño ejemplar, un desempeño adecuado, en desarrollo, e insuficiente.

Para el desarrollo de una rúbrica, se debe tener en consideración los siguientes pasos:

1. Determinar los objetivos del aprendizaje.
2. Identificar los aspectos a evaluar.
3. Definir descriptores, escalas y además los criterios de evaluación (Figuras 2).
4. Determinar un valor para cada criterio.
5. Revisar la rúbrica y evaluar su impacto en los educandos.

Conceptos/rubros	Escalas/niveles ejecución (cuantitativo/cualitativo/mixto)			
	4	3	2	1
Aspectos a evaluar	Criterios evidencias a alcanzar	Criterios evidencias a alcanzar	Criterios evidencias a alcanzar	Criterios evidencias a alcanzar

Figuras 2: Ejemplo de Rúbrica de Evaluación

3.3 Software, Tecnologías, Técnicas y abreviaciones.

Todo el Software utilizado para el desarrollo del proyecto de Software se encuentra con licencia gratuita o cuenta con un programa para estudiantes, en el cual entregan una licencia para su uso por un tiempo determinado.

En la siguiente sección se encuentra un listado del Software utilizado (y algunas de sus alternativas) y el enlace a su respectiva descarga o programa para estudiantes según corresponda.

3.3.1 Entornos de desarollo:

- **IntelliJ Idea**, Distribuido de manera Gratuita para estudiantes, con licencia renovable de manera anual en el siguiente enlace: https://www.jetbrains.com/student/
- **Spring Tool Suite**, IDE para Spring basado en Eclipse, distribuido de manera gratuita en: https://spring.io/tools

3.3.2 Herramientas para la gestión del proyecto

- **Trello**, Cuenta que provee de servicios gratuitos para la administración de un tablero ágil en el enlace: https://trello.com

3.3.3 Sistemas de control de versiones:

- **GitHub**, Cuenta de desarrollador entregada de manera gratuita por dos años en el siguiente enlace: https://education.github.com/pack/offers
- **Git Kraken**, Cuenta de estudiante por un año, al enlazar una cuenta de estudiante en GitHub, para obtener utilizar el enlace: https://education.github.com/pack/offers
- **Git SCM**: Versión de consola para utilizar Git, distribuido de manera gratuita en: https://git-scm.com

3.3.4 Herramientas para el despliegue

- **FileZilla**, distribuido de manera gratuita en el siguiente enlace: https://filezilla-project.org/download.php?type=client

3.3.5 Herramientas para la interacción con la aplicación

- **Postman**, distribuido de manera gratuita en el siguiente enlace: https://www.getpostman.com/

La disponibilidad de todas estas herramientas fue comprobadas por última vez en noviembre del 2017.

Además de esto, en el Anexo 3 y 4, se encuentra una definición de todas Las tecnologías, técnicas y abreviaciones utilizadas a lo largo de este documento.

4 ESTRUCTURA DEL MATERIAL DIDACTICO

En esta sección se hará entrega de los detalles sobre el material teórico-práctico, su contenido, la estructura general, y el tiempo estimado para el desarrollo de cada una de las unidades presentadas.

4.1 Elección de los Contenidos.

Dentro del presente trabajo se nombraron diferentes tipos de metodologías, técnicas y tecnologías, las que fueron escogidas por diferentes motivos, en la siguiente sección se presentara las razones de esta selección.

- **Spring Boot**: Este framework fue seleccionado como la tecnología a utilizar, por la gran cantidad de ofertas laborales que está recibiendo hoy en día y además que se acomoda bien a la malla curricular actual, ya que esta está fuertemente enlazada a utilizar el lenguaje Java, el cual además es uno de los lenguajes más populares en la industria según el TIOBE Index (TIOBE Index).

- **Scrum:** Esta metodología es la elección más común de las metodologías de desarrollo ágil, por lo que es altamente demandado tener conocimiento y experiencia de está. (Schwaber & Sutherland, 2017)

- **Git:** Este Sistema de Control de Versiones (SCM por su sigla en inglés), es la herramienta para la gestión de proyectos más popular hoy en día, la fuerza que cobraron plataformas como GitHub, dejan demostrado de manera empírica la necesidad de conocer dicha herramienta[8].

- **Test Driven Development:** esta una técnica de desarrollo basada en el uso de Test Unitarios, la cual a medida que acumula Test el proyecto va creciendo, por lo que se propone como una forma de desarrollar en la cual los estudiantes puedan confiar en sus compañeros de equipo (Jurado, 2010).

[8] https://github.com/about

- **Code Review**: esta una técnica que se utiliza en metodologías agiles para poder exponer el código de un programador al resto del equipo, y obtener diferentes opiniones sobre este, relacionadas con patrones de diseño, redundancia de código, etc. La idea de utilizar esta técnica es que los estudiantes puedan compartir sus conocimientos dentro de los equipos de trabajo[9].

Para la definición del contenido del material de estudio se utilizó la información obtenida desde la sección III del esquema extenso de la asignatura en la cual se menciona que los estudiantes deben "Aplica de manera sistemática métodos, técnicas, herramientas y estándares de requerimientos, análisis y diseño para desarrollar un software que satisfaga los requerimientos técnicos, funcionales y no - funcionales.", además se consideró la demanda de las diferentes tecnologías, metodologías y técnicas dentro de la industria en diferentes sitios de empleo como GetOnBrd[10].

4.2 Estructura general del material.

El material presentado cuenta de cuatro partes por cada una de las unidades presentadas en la sección anterior, las cuales están distribuidas de la siguiente forma:

- **Material de estudio**: corresponde a lecturas, las cuales presentan el contenido de la unidad, las que pueden ser entregadas al estudiante para complementar su trabajo autónomo.

- **Material de trabajo**: corresponde a actividades tanto individuales tanto como grupales, para el desarrollo presentar el contenido del material de estudio de forma práctica, estas están desarrolladas para fomentar el trabajo autónomo del estudiante.

- **Autoevaluación**: corresponde a una sección que contiene una serie de ejercicios con los cuales el estudiante puede medir sus conocimientos y evaluarse según la rúbrica presentada en las distintas secciones.

[9] https://www.atlassian.com/agile/software-development/code-reviews
[10] https://www.getonbrd.cl/

- **Referencias sobre la unidad**: corresponde a las fuentes consultadas para la construcción del material, también incluye lecturas recomendadas para profundizar el conocimiento sobre su sección correspondiente.

Dentro de estas partes, en las unidades de estudio y material de trabajo, podremos encontrar, texto, imágenes, lecturas recomendadas, video tutoriales y otros elementos que sean a fin con la unidad correspondiente.

4.3 Tabla de Contenidos.

En el siguiente recuadro se detallara los contenidos de cada una de las unidades y el tiempo esperado para el desarrollo de estas.

Contenido	Objetivos principales	Tiempo esperado
Introducción a la metodología ágil Scrum.	Presentar a los estudiantes la metodología de desarrollo Scrum con ejemplos de su estructura y funcionamiento.	4 Horas Autónomas.
Estimación de historias de usuario.	Presentar a los estudiantes un ejemplo práctico de la actividad Planning Poker para la estimación de historias de usuario.	4 Horas Autónomas.
Gestión de proyectos con Git.	Enseñar a los estudiantes el uso de un sistema de control de versiones.	4 Horas Autónomas.
Code Review y Test-Driven Development	Estudiar las técnica para el desarrollo de software Code Review y Test-Driven Development y lo que esta conlleva	8 Horas Autónomas.
Spring Boot y los Servicios REST	Presentar el Framework Spring como una opción para la construcción de Software y la importancia de los servicios REST.	10 Horas Autónomas.

Desarrollo de un Servicio Web del tipo REST con Spring Boot	Desarrollar un Web Service utilizando los contenidos vistos anteriormente y además deberán realizar la construcción mediante el uso de TDD.	10 Horas Autónomas.

Tabla 1: Contenido Unidades Material Didáctico.

5 CONSTRUCCIÓN DEL MATERIAL

En esta sección se darán a conocer los objetivos del producto y una descripción de este, para luego dar a conocer los detalles de la construcción del material didáctico presentado el seguimiento del Product Backlog presentado en el tablero ágil a lo largo de los distintos incrementos presentados.

5.1 Objetivos del producto

Proporcionar un material de estudio, el cual permita a los estudiantes adquirir los conocimientos para el desarrollo de un proyecto de software por el lado de Backend utilizando diferentes técnicas y metodologías para el desarrollo de software, y además presentar un prototipo que sirva de guía para apoyar la construcción de dicho software por parte de los estudiantes.

5.1.1 Objetivos generales por Unidad.

1. **Introducción a la metodología ágil Scrum.**

   Durante el desarrollo de este capítulo se espera que los estudiantes comprendan la estructura y funcionamiento de una de las metodologías agiles más populares dentro del ámbito del desarrollo de Software.

2. **Estimación de historias de usuario.**

   Durante el desarrollo de este capítulo se espera que los estudiantes sean capaces de poder realizar la topa de historias de usuario de forma sencilla, y puedan estimar el rendimiento de su equipo acorde a sus capacidades que ellos estiman.

3. **Gestión de proyectos con Git.**

   Durante el desarrollo de este capítulo se espera que los estudiantes puedan ser capaces de utilizar herramientas para el versionamiento del código fuente de sus aplicaciones y distribuir el trabajo entre los miembros del equipo.

4. **Code Review y Test-Driven Development**

   Durante el desarrollo de este capítulo se espera que los estudiantes obtengan conocimiento sobre diferentes técnicas de programación para aplicar en el desarrollo del proyecto de la asignatura, en donde aprendan los beneficios del trabajo en equipo y la utilización de herramientas para desarrollar Test.

5. **Spring Boot y los Servicios REST**

   Durante el desarrollo de este capítulo se espera que los estudiantes aprendan los beneficios del uso del framework Spring Boot y puedan realizar sus propias aplicaciones utilizando dicha plataforma, además, se espera que los estudiantes puedan adquirir los conocimientos necesarios para implementar una API Rest siguiendo el Rest Maturity Model.

6. **Desarrollo de un Servicio Web del tipo REST con Spring Boot**

   Durante el desarrollo del siguiente capítulo, se presentará una problemática sencilla, a la cual se le dará solución utilizando parte de las metodologías, técnicas y tecnologías presentadas a lo largo de este documento.

5.2 Definición de los requerimientos en un tablero ágil

Durante esta sección se presentaran las primeras tareas detectadas según los requerimientos del proyecto propuesto y como a estas se les asignó un tamaño determinado para tener un estimado respecto al esfuerzo de desarrollo del proyecto.

1. Encuestar a los estudiantes que hayan rendido a la asignatura de Ingeniería de Software, para encontrar necesidades sobre su aprendizaje.
2. Redactar un capítulo de material de estudio respecto a la agilidad y a la metodología de desarrollo Scrum
3. Redactar un capítulo de material de estudio respecto al uso de Historias de usuario y como estimar con estas el tiempo del proyecto
4. Redactar un capítulo de material de estudio referente al uso de Sistemas de Control de Versiones (Git).
5. Redactar un capítulo de material de estudio que tenga relación con el uso de las técnicas para el desarrollo de Software TDD y Code Review.

6. Redactar un capítulo de material de estudio respecto a los Framework Spring Boot y los servicios REST.
7. Preparar un Ejemplo de desarrollo utilizando Historias de Usuario, Spring Boot y herramientas para auto documentar.

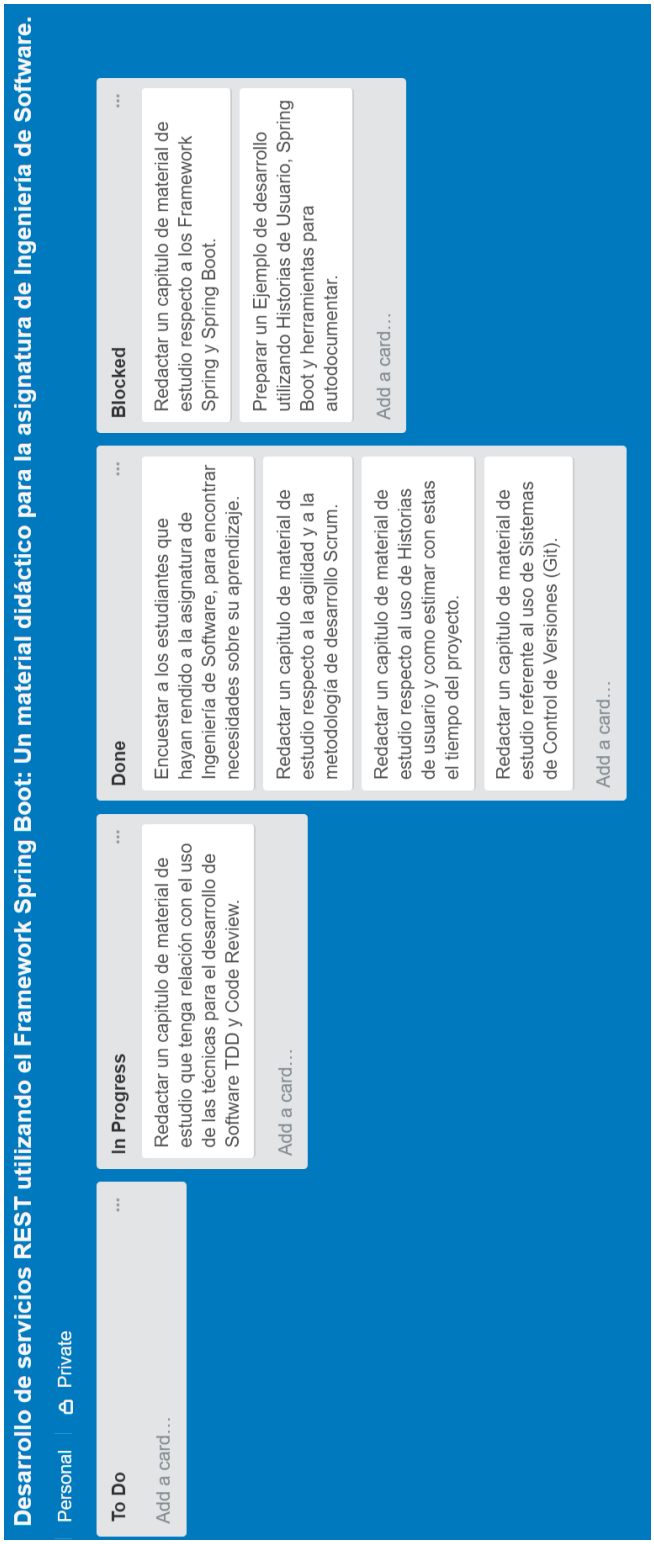
Para la producción del material de trabajo y estimar el tamaño de las tareas se utilizó la técnica de comparar según la historia más pequeña conocida, en la cual se toma la historia de usuario más pequeña que el equipo de trabajo conozca y comienza a estimar según el esfuerzo considerado para la realización de dicha tarea, además, como unidad de medida se utilizaron tallas de ropa, por lo cual, las medidas de las historias van entre S, M, L, y XL.

En este caso en particular, se utilizó la historia número dos, la cual corresponde a la preparación de un material de estudio que respecto a la agilidad y la metodología de desarrollo Scrum, ya que esta historia, servirá como una guía para el resto de las historias directamente relacionadas con el desarrollo de más material didáctico para el producto final. A esta tarea se le asignó un tamaño M, por la duración de esta y la cantidad de revisiones que recibió, por lo que las otras tareas relacionadas con el desarrollo del material didáctico tuvieron al menos una talla M, la cual se ve reflejada en la siguiente tabla:

Tarea	Talla
Toma de Encuestas	S
Agilidad	M
Historias de Usuario	M
Git	L
Spring Boot y Servicios Rest	L
Proyecto Piloto	M
Encuestas para cierre	S

Tabla 2: Asignación de tamaños en las historias.

En la siguiente Figura, se presenta un ejemplo de cómo se visualizan las tareas en el Software de Gestión Trello, en la cual se pueden distribuir estas en diferentes estados según sea requerido en el tablero utilizado.

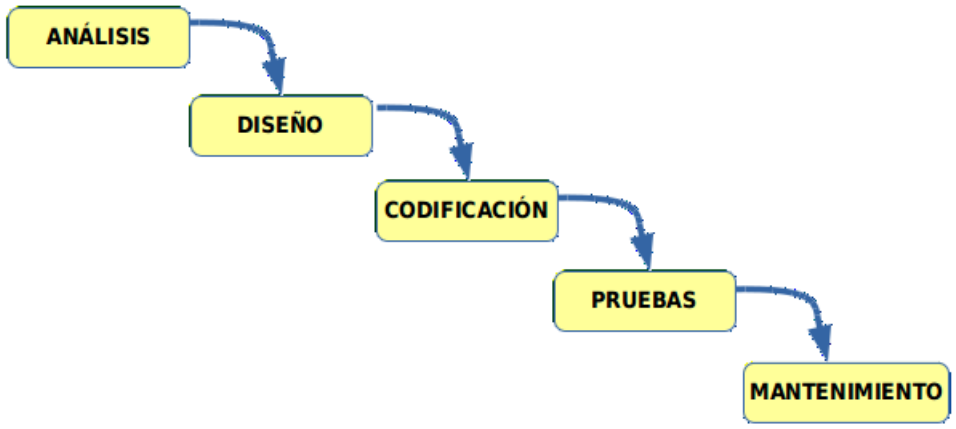


Figuras 3: Tablero Trello

Para el desarrollo de las siguientes historias, se siguió un modelo similar al de Scrum, como el que se puede ver en la siguiente figura:

Figuras 4: Modelo Scrum[11]

En el cual se iban tomando las diferentes historias para procesarlas como una tarea durante un periodo de tiempo determinado (aproximadamente 4 a 6 semanas) para dichas tareas según su esfuerzo estimado, en donde las revisiones con el cliente o los stakeholders (Estudiantes, Docentes de otras asignaturas, Profesionales) eran bastante esporádicas asemejando el proceso de desarrollo de cada historia a una pequeña iteración de la metodología cascada, en la siguiente figura se puede visualizar un modelo clásico de desarrollo en cascada:



Figuras 5: Modelo de desarrollo en cascada[12]

[11] https://lorbada.com/es/diferentes-metodologias-agiles
[12] http://jamj2000.github.io/entornosdesarrollo/1/diapositivas#/2/11

Esta metodología de desarrollo fue sutilmente modificada para integrar el uso de las historias de usuario como se especificó anteriormente, y además la sección de pruebas se realizaba cada cierto tiempo con estudiantes del grupo que respondió la encuesta mencionada en el Anexo 1. Dichos estudiantes entregaban feedback sobre el material, y si este cumplía con las expectativas del apoyo que ellos les hubiese gustado tener a la hora de rendir la asignatura y realizar el portafolio de proyecto de software de esta.

Esto además, sirvió para definir un alcance sobre la profundidad del material y permitir solo generar una especie de introducción al framework Spring Boot y de esta forma suavizar la curva de aprendizaje de este para los estudiantes.

6 DESCRIPCIÓN DE LOS PROTOTIPOS Y SOLUCIONES PARCIALES

Durante esta sección se describen el proceso de la construcción de algunas de las soluciones presentadas en el "Material Didáctico", en las cuales era requerido construir un software durante el proceso.

6.1 Frameworks de Prueba JUnit y Mockito

Los Framework de pruebas dentro del contexto del material didáctico juegan un papel importante, ya que estos le permitirán al desarrollador implementar la técnica para la construcción de Software llamada Test-Driven Development, descrita previamente en la sección 3.

6.1.1 JUnit

En el contexto en el cual el framework de pruebas unitarias JUnit fue utilizado dentro de este trabajo, permitió hacer el uso de la técnica de desarrollo llamada Test-Driven Development, durante la resolución de los diversos ejercicios propuestos en el material de estudio.

6.1.2 Mockito

En este mismo contexto el Framework Mockito, funciona como parte de JUnit y permitirá simular instancias de una clase en específico, con lo que podremos crear resultados para las clases ajenas al módulo del negocio que realmente queremos probar y de esta forma poder aislar los Test de las distintas unidades de negocio.

6.2 Gradle, como gestor de dependencias

Gradle es un gestor de dependencias, basado en las ideas originales de Apache Ant y Apache Maven, el cual utiliza su propio lenguaje basado en Groovy en vez de los XML utilizados por Maven, para declarar la configuración del proyecto.

Durante el desarrollo de este trabajo se utilizó como fuente el MVNRepository1[13] para descargar las dependencias necesarias para cada uno de los ejercicios resueltos.

[13] https://mvnrepository.com/

6.3 Coding Katas

Las Coding Katas son ejercicios de programación en los cuales se le propone al programador un pequeño problema y una lista de requerimientos para implementar la solución de este, la idea de las Coding Kata es repetir los ejercicios de forma que el desarrollador logre interiorizar los conceptos que está trabajando mediante la repetición.

Este tipo de ejercicios probablemente fueron originados dentro del libro "*The Pragmatic Programmer*", en un intento de asimilar el concepto de las katas de las artes marciales en el mundo de la programación.

La principal función de estos a lo largo del material de estudio es que el estudiante se vea en la obligación de mejorar sus habilidades técnicas mediante este tipo de ejercicios.

6.3.1 Fizz Buzz Kata

Es un ejercicio bastante popular dentro de las entrevistas técnicas dentro de la industria, en donde se expone una problema sencillo en del cual se espera obtener información respecto a las habilidades del postulante a la hora de detectar y tratar un caso borde en específico.

Este ejercicio normalmente provee las siguientes instrucciones:

Escriba un programa el cual cumpla lo siguiente:

Imprima los números del 1 al 100 pero,

- *Cuando el número sea múltiplo de 3, imprima Fizz.*
- *Cuando el número sea múltiplo de 5, imprima Buzz.*
- *Y finalmente, cuando el número sea múltiplo de 3 y 5, imprima FizzBuzz.*

Este ejercicio, fue adaptado como una Coding Kata en la cual el desarrollador deberá implementar un método en el cual según una determinada entrada deba retornar el valor esperado.

Para lograr este objetivo, cada uno de los diferentes requerimientos deberá ser implementado como un Test Unitario para este caso JUnit, además, una solución propuesta para este ejercicio fue anexada al material didáctico la cual se presenta en las siguientes imágenes:

```java
public String evaluate(int value) {
    String fb = Integer.toString(value);
    if (value % 3 == 0) {
        fb = "fizz";
    }
    if (value % 5 == 0) {
        fb = "buzz";
    }
    if (value % 5 == 0 && value % 3 == 0){
        fb = "fizzbuzz";
    }
    return fb;
}
```

Figuras 6: Implementación Solución

Para lograr esta implementación para la solución de este ejercicio se hizo el seguimiento a una bateria de test que permitieran comprobar que el funcionamiento de dicho metodo cumplía con las condiciones presentadas en el enunciado.

```java
@Test
public void shouldBeAbleToEvaluate1andReturn1() {
    String one = fizzBuzz.evaluate( value: 1);

    assertEquals( expected: "1", one);
}

@Test
public void shouldBeAbleToEvaluate2AndReturn2() {
    String two = fizzBuzz.evaluate( value: 2);

    assertEquals( expected: "2", two);
}

@Test
public void shouldBeAbleToEvaluate3AndReturnFizz() {
    String fizz = fizzBuzz.evaluate( value: 3);

    assertEquals( expected: "fizz", fizz);
}
```

Figuras 7: Primeras pruebas

Durante estas pruebas se puede verificar que el método está retornando lo esperado, para los primeros casos que fueron considerados.

```
@Test
public void shouldBeAbleToEvaluate10AndReturnBuzz() {
    String buzz = fizzBuzz.evaluate( value: 10);

    assertEquals( expected: "buzz", buzz);
}

@Test
public void shouldBeAbleToEvaluate15AndReturnFizzBuzz() {
    String fb = fizzBuzz.evaluate( value: 15);

    assertEquals( expected: "fizzbuzz", fb);
}
```

Figuras 8: Casos de prueba

En los presentes casos, se comprobó, que además era capaz de manejar el caso en que el valor era múltiplo de 5 y además se comprobó el funcionamiento del caso borde, en el que el valor es múltiplo de 3 y 5.

6.3.2 Stack Kata

Este ejercicio está basado en el ejercicio expuesto en "Code Katas: The Stack" publicada en el blog de "Jack Reichert, durante este ejercicio se le solicita al desarrollador crear la estructura de datos de la pila mediante TDD, en la cual se verá enfrentado a diferentes casos bordes en su avance por el código, lo que implicará que deberá implementar nuevas funciones, borrar código antiguo, refactorizar el código

Durante este ejercicio originalmente se requiere de lo siguiente:

- Puedes crear un objeto del tipo Stack.
- Todo Stack recién creado, deberá estar vacío.
- Despues de hacer push una vez, el tamaño del Stack deberá ser uno.
- Luego de un push y un pop, el Stack deberá estar vacío.
- Cuando haces push sobre un límite, el Stack deberá lanzar una excepción de overflow.
- Cuando haces pop sobre un Stack vacío, el Stack deberá lanzar una excepción de underflow.
- Cuando dos se hace push a dos valores, y luego se realiza un pop, el tamaño, deberá ser uno.

- Cuando se hace push a un elemento, el siguiente elemento recibido al hacer pop, debe ser el mismo.
- Cuando se hace push a dos elementos, se deben retornar en el orden inverso al hacer dos pop.
- Al crear un stack, considera un constructor, si el valor del tamaño máximo es negativo, debes lanzar una excepción del tipo IllegalCapacity.
- Cuando se crea un Stack con capacidad 0, cualquier push deberá lanzar overflow.
- Cuando se pasa un valor por push, el mismo valor estará en top.
- Cuando el Stack está vacío, y se usa top, deberá devolver un vacio.
- Cuando la capacidad es 0, top siempre retornara vacio.

En el repositorio del siguiente repositorio se puede encontrar una solución parcial de este ejercicio: https://github.com/eternaletulf/stackkata.

Dentro de dicho repositorio al ver los explorar los cambios en los diferentes commits se puede observar como el código evolucionaba con cada iteración de TDD, logrando hacer que el código final de producción cumpliese con lo requerido en un principio.

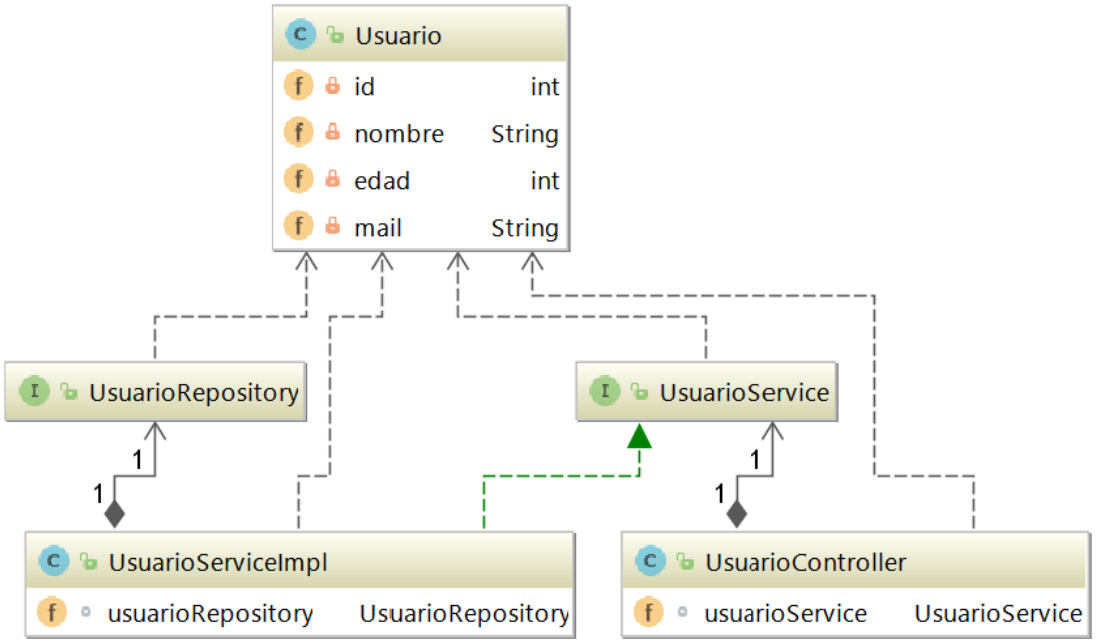
6.4 Spring Boot

El Framework Spring Boot está basado en Spring Core, el cual fue escrito por Rod Johnson el año 2002, el cual tiene como objetivo facilitar la construcción de aplicaciones del tipo empresarial usando el lenguaje Java, trayendo consigo una amplia cantidad de paquetes diseñados para facilitar el trabajo en este contexto.

Dentro del material didáctico presentado, en la sección que corresponde a "Spring Boot y los Servicios REST", se le solicita a los estudiantes construir un pequeño *Spike*, utilizando este framework en el cual construyan un pequeño servicio REST el cual permita realizar las operaciones básicas de un CRUD (Create, Read, Update, Delete) en una determinada entidad llamada Usuario, la que está conformada por los siguientes atributos:

- Identificador,
- Nombre,
- Edad
- Correo.

Por lo que del desarrollo de dicha actividad los estudiantes deberían ser capaces de desarrollar un sistema bastante sencillo en el cual se cumpla con lo propuesto anteriormente por lo que tendrán un diagrama de clases de su aplicación bastante similar al de la siguiente figura:



Figuras 9: Diagrama UML del Spike

En donde la Entidad Usuario, como nuestro POJO debera relacionarse con el resto de las clases del sistema.

Nuestro UsuarioRepository, servira como una conexión a nuestra base de datos H2 gracias a las facilidades que entrega Spring Boot.

UserService, nos permitirá tener una Interfaz que describirá el funcionamiento de UserServiceImpl que es la clase en donde se alojará el codigo core del negocio, en este caso se encargara de hacer las llamadas al CRUD.

UserController, sera la clase que permitirá la interacción del sistema con un usuario externo mediante diferentes URIs.

```java
@Entity
public class Usuario {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String nombre;
    private int edad;
    private String mail;
```

Figuras 10: POJO Usuario

Dentro de este POJO se encuentran las operaciones básicas de un objeto, como su constructor por defecto (constructor vacío) y además se encuentran los getters y setters de este.

```java
public interface UsuarioService {

    Usuario findById(int id);

    List<Usuario> findAll();

    Usuario save(Usuario usuario);

    Usuario removeById(int id);
}
```

Figuras 11: Interfaz de la capa de Servicios

Dentro de esta interfaz se presentan las diferentes funciones que debe tener las clases que implementen los servicios para este Usuario, por lo que le servirá como una guía al desarrollador, para poder implementar dichos servicios.

```java
@Service
public class UsuarioServiceImpl implements UsuarioService {

    @Autowired
    UsuarioRepository usuarioRepository;

    @Override
    public Usuario findById(int id) {
        return usuarioRepository.findById(id);
    }

    @Override
    public List<Usuario> findAll() {
        return usuarioRepository.findAll();
    }

    @Override
    public Usuario save(Usuario usuario) {
        return usuarioRepository.save(usuario);
    }

    @Override
    public Usuario removeById(int id) {
        Usuario usuario = usuarioRepository.findById(id);
        usuarioRepository.removeById(id);
        return usuario;
    }
}
```

Figuras 12: Capa de Servicios para Usuario

Dentro de esta clase de Servicios se implementa la interfaz recién estudiada, dando paso a que el compilador de Java obligue al desarrollador a implementar todos los métodos de esta, por lo que de esta clase se realizaran las llamadas a la base de datos accediendo a esta mediante a la instancia que Spring genero mediante el uso de la etiqueta *@Autowired* de UsuarioRepository.

```java
@Repository
public interface UsuarioRepository extends JpaRepository<Usuario, Integer> {

    Usuario findById(int id);

    void removeById(int id);

    Usuario save(Usuario usuario);

}
```

Figuras 13: Interfaz Repository

Mediante el uso de esta interfaz, que extiende de JpaRepository, Spring realiza mediante un ORM las operaciones necesarias para poder interpretar cada uno de los métodos como una consulta para la base de datos, en memoria H2.

El ORM que implementa Spring es Hibernate, y este procura transformar desde la firma de los métodos en la interfaz en la que se extiende, y además manejar la conexión a la base de datos.

Dentro de su implementación, Hibernate realiza las operaciones mediante JDBC el cual recibe las consultas creadas desde estas conversiones, y maneja la conexión a la base de datos, para luego recibir un ResultSet de JDBC y que se transformar en el POJO (Plain Old Java Object) que corresponda.

```java
@RestController
@RequestMapping("/usuario/")
public class UsuarioController {

    @Autowired
    UsuarioService usuarioService;

    @GetMapping()
    public ResponseEntity<List<Usuario>>findAll(){
        return new ResponseEntity<>(usuarioService.findAll(), HttpStatus.OK);
    }

    @GetMapping("{id}")
    public ResponseEntity<Usuario>findById(@PathVariable("id") int id){
        return new ResponseEntity<>(usuarioService.findById(id), HttpStatus.OK);
    }

    @PostMapping
    public ResponseEntity<Usuario> save(@RequestBody Usuario Usuario){
        usuarioService.save(Usuario);
        return new ResponseEntity<>(Usuario, HttpStatus.ACCEPTED);
    }

    @PutMapping
    public ResponseEntity<Usuario> update(@RequestBody Usuario Usuario){
        usuarioService.save(Usuario);
        return new ResponseEntity<>(Usuario, HttpStatus.ACCEPTED);
    }

    @DeleteMapping("{id}")
    public ResponseEntity<Usuario> delete(@PathVariable("id") int id){
        return new ResponseEntity<>(usuarioService.removeById(id), HttpStatus.OK);
    }
```

Figuras 14: Clase Controladora

Esta es la clase que finalmente da con el usuario final, en la cual se permite la comunicación por sobre el protocolo HTTP, y las comunicación se hace mediante REST, de la cual se proveen recursos en sus respectivas URI (*Unified Resource Identifier*), que están representados mediante respuestas que incluyen un JSON y una respuesta HTTP. Así también esta clase para obtener los resultados, se conecta a la clase de servicios mediante su interfaz, dejando que Spring pase una instancia de la clase que implementa dicha interfaz.

6.5 Solucíon del prototipo final del material de estudio

Dentro del material de didáctico se propone el desarrollo de un proyecto sencillo en el cual se requiere del uso de todas las técnicas, metodologías y tecnologías aprendidas a lo largo de la asignatura, por lo que el estudiante deberá poner a prueba todos los conocimientos adquiridos a lo largo del curso.

6.5.1 Enunciado

Considere que usted y su equipo han sido contratados para resolver el siguiente problema en la cual deben usar Spring Boot para crear un servicio del tipo REST:

Una compañía de telecomunicaciones le solicita a usted crear un sistema que le permita añadir, modificar, eliminar, y obtener los registros de sus clientes y los servicios que cada uno de ellos tenga asociado. Entre los servicios que ofrece esta empresa pueden ser de televisión y/o internet, los cuales se proveen mediante diferentes tipos de planes.

La compañía ofrece dos planes de televisión diferentes, dos planes de internet, y dos tipos de pack los que incluyen televisión e internet, los planes que ofrece esta compañía están descritos en la siguiente tabla:

ID	Plan	Características	Precio
TVIT-4040	Conexión total	Incluye internet de 40Mbps y 40 Canales.	$40.000.-
TVIT-4020	Muy conectado	Incluye internet de 40Mbps y 20 Canales.	$30.000.-
IT-40	Internet Turbo	Incluye internet de 40Mbps.	$25.000.-
IT-20	Internet	Incluye internet de 20Mbps.	$15.000.-
TV-40	Televisión 40	Incluye 40 Canales.	$25.000.-
TV-20	Televisión 20	Incluye 20 Canales.	$10.000.-

Tabla 3: Oferta de planes

Actualmente la información de los clientes solo se almacena en un archivo el cual tiene una estructura que corresponde al siguiente formato para cada uno de sus clientes:

RUN	Nombre Completo	Teléfono	Ciudad	Dirección	Mail
1.111.111-1	Juan Turbo Lezca	123456789	Chillan	Dir….	mail@mail.com

Tabla 4: Formato de almacenado de clientes

Tenga en cuenta, que la única información que contiene el contrato, son: el número de contrato, los datos del cliente, el tipo de plan contratado y la fecha de contrato.

Considere que la empresa ya cuenta con un equipo de desarrollo para el Front end, por lo que usted y su equipo solo deberán concentrar sus esfuerzos en entregar un servicio del tipo REST que permita solventar los requerimientos del cliente.

Entorno de Desarrollo recomendado para la implementación de la solución.

Para el problema presentado se espera a que los estudiantes involucrados puedan utilizar la metodología Scrum para la gestión del proyecto, de forma que los estudiantes puedan conformar el rol de Development Team y que el docente a cargo de la asignatura pueda suplir los roles tanto de Scrum Master y Product Owner, de forma que el docente pueda velar por el cumplimiento de las distintas ceremonias de la metodología y la correcta interpretación de los requerimientos mediante historias de usuario.

Recomendaciones para el entorno de desarrollo:

Para el desarrollo del siguiente proyecto se recomienda el uso de las siguientes herramientas:

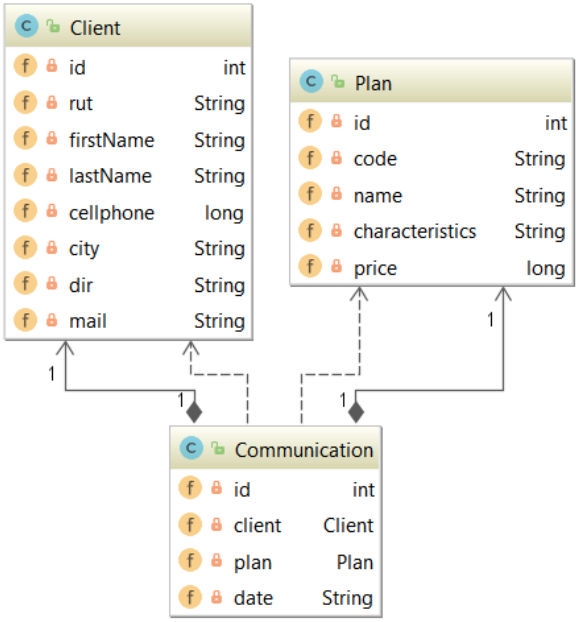
- **Trello**, para la gestión del Kanban.
- **GitKraken o Git Bash**, para el manejo del control de versiones y seguimiento del proyecto.
- **IntelliJ Idea**, para el desarrollo del proyecto.
- **Mockaroo,** para la poblar de datos la base de datos.
- **JDK 1.8 o superior,** para poder utilizar el framework Spring Boot.

Como comento en la sección 3, estas herramientas están disponibles de forma gratuita o con licencia para estudiantes, en los enlaces presentados.

Así mismo, se recomienda el uso de diferentes técnicas para el desarrollo como Test-Driven Development, y que como grupo puedan escoger una normativa para la escritura de su código fuente un ejemplo de esto es la Google Style Guide para Java.[14]

6.5.2 Descripción de la solución

Dentro del contexto de este prototipo, lo primero que se puede identificar según el enunciado presentado son las entidades que son nombradas y descritas de manera explícita en tablas y/o texto.



Figuras 15: Modelo de datos

Modelo de datos el cual es traducido automáticamente por Hibernate, permitiendo que dichos objetos sean reflejados en tablas en una base de datos relacional mediante este ORM[15].

En cuanto a lo que respecta a las historias de usuario, gran parte de estas se pueden identificar de manera rápida, ya que el enunciado presenta claramente las intenciones del cliente solicitando las operaciones del CRUD en el segundo párrafo.

[14] https://google.github.io/styleguide/javaguide.html
[15] http://hibernate.org/

Por lo que tendremos historias tales como:

- Como usuario quiero saber los datos de un Cliente en específico, para saber cuánto tiempo lleva con nosotros en la compañía.
- Como usuario quiero saber cuáles son las ofertas de planes disponibles tanto para televisión e internet.
- Como usuario, quiero poder ver a todos los Clientes registrados en el sistema, para poder tener información clara sobre la situación actual.
- Como usuario, deseo poder eliminar a los Clientes que ya no tengan planes asociados para evitar problemas en el futuro.
- Como usuario, quiero poder ver los detalles de un contrato según un ID en específico para poder saber a qué cliente corresponde y que plan tiene.
- Como usuario, quiero poder cambiar la oferta de planes según mis necesidades.
- Como usuario, quiero poder actualizar los datos de un cliente de ser necesario.

Dicho esto, se consideró lo siguiente en el desarrollo de la solución del prototipo:

- La creación de pruebas unitarias para cubrir al menos la capa de servicios, en un 80%, la capa de controladores no es exigida, ya que este tipo de Test no fue cubierto durante la construcción del material didáctico, *pero aun así la solución completa incluye ejemplos de este tipo de Test unitarios*.
- Integración de Swagger para auto-documentar la aplicación.
- El uso de una base de datos en memoria (H2).
- La puesta en producción de la aplicación dentro del servidor de la Universidad.

Repositorio de la solución: https://github.com/eternaletulf/ProyectoDeTitulo

6.5.3 Paso a producción

Para el paso a producción de la aplicación es necesario contar con una maquina la cual permita desplegar el proyecto, para esto es necesario tener una máquina que cuente con un web container para aplicaciones Java.

Para esto caso, se desplegará la aplicación en un servidor de la universidad, para lo cual se debe solicitar el espacio en los mediante el sistema de laboratorios de especialidad, alojado en el

servidor Arrau[16]. Al tener las credenciales de acceso a este se deberá conectar a este servidor mediante el uso del protocolo SFTP, siguiendo las instrucciones provistas dentro del sistema de laboratorios utilizando un cliente como FileZilla para poder subir el archivo correspondiente.

Dentro del espacio reservado se encontrará una carpeta con nombre "war", dentro de esta se deberá ubicar el archivo empaquetado (o Artefacto de Software), el cual deberá tener el nombre de usuario que le fue asignado y además el empaquetado deberá tener el formato "war.

Para poder cumplir con lo mencionado anteriormente y poder realizar el despliegue de la aplicación, se debe verificar y realizar lo siguiente:

En el archivo ***build.gradle*** en la raíz del proyecto se debe verificar que la dependencia que corresponde a Tomcat debe estar configurada como "providedRuntime" ya que esta aplicación va a ser montada en un web container Tomcat el cual está alojado en Colvin, y de esta forma se evitará que ocurran conflictos ya que la aplicación no intentará levantar su propia instancia de Tomcat.

```
dependencies {
    compile('org.springframework.boot:spring-boot-starter-data-jpa')
    compile('org.springframework.boot:spring-boot-starter-web')
    compile("org.springframework.boot:spring-boot-devtools")
    /* compile('org.springframework.boot:spring-boot-starter-tomcat')
    Para correr en el servidor Colvin, cambiar la dependencia a providedRuntime*/
    providedRuntime('org.springframework.boot:spring-boot-starter-tomcat')
    runtime('com.h2database:h2')

    compile group: 'io.springfox', name: 'springfox-swagger2', version: '2.6.1'
    compile group: 'io.springfox', name: 'springfox-swagger-ui', version: '2.6.1'

    testCompile('org.springframework.boot:spring-boot-starter-test')
}
```

Figuras 16: Dependencias build.gradle del prototipo

En el siguiente paso se ejecuta Gradle mediante la línea de comandos usando como parámetro "build" de forma que cree un archivo ejecutable, y por la naturaleza del proyecto al ser Web deberá ser un archivo del tipo WAR, el cual se encuentra dentro de la carpeta */build/lib/* el cual Gradle creará en el directorio de raíz de la aplicación.

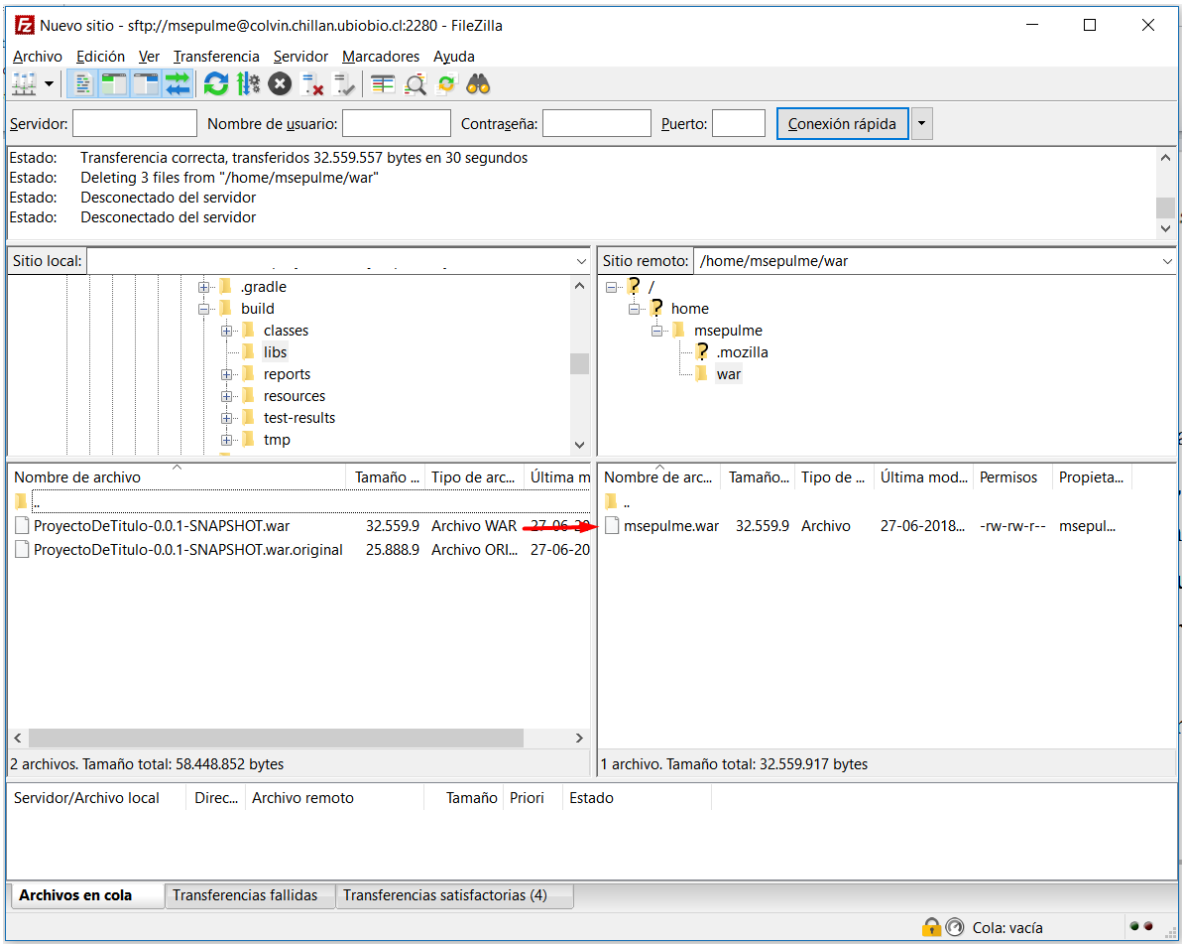
[16] http://arrau.chillan.ubiobio.cl/laboratorio/index.php

```
Terminal
Microsoft Windows [Versión 10.0.17134.112]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

D:\projects\ProyectoDeTitulo>./gradlew build
```

Figuras 17: Construcción del artefacto

Luego se deberá utilizar FileZilla y lograr una conexión al servidor mediante el uso del instructivo que se encuentra preparado de antemano en el Sistema de Laboratorios de Especialidad, luego de establecer la conexión se deberá copiar el archivo de extensión WAR de la carpeta local en el servidor y además deberá cambiar el nombre del archivo, por el nombre de usuario como se muestra en la siguiente figura:



Figuras 18: Subiendo archivos al servidor Arrau con FileZilla

Luego de esto se deben esperar unos 5 minutos para que la aplicación esté disponible, de haber algún error en el proceso, la aplicación no arrancará y podrá solicitar los *LOGs* al encargado del servidor.

6.5.4 Muestra en producción

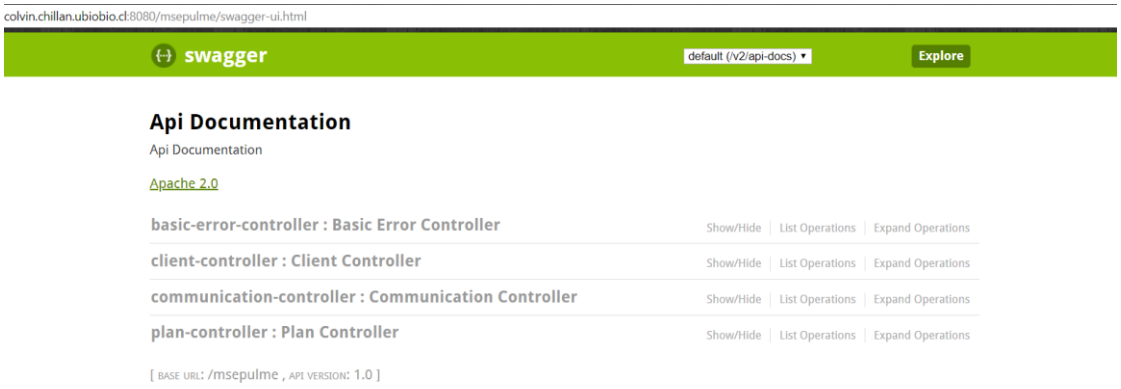
Actualmente, se encuentra disponible en el servidor la última versión de este prototipo la que se encuentra funcionando dentro del servidor Colvin de la universidad, ubicado en el siguiente URL:

http://colvin.chillan.ubiobio.cl:8080/msepulme/

De la cual se pueden obtener respuestas de los diferentes servicios REST provistos por esta aplicación, así mismo se cuenta con la implementación de Swagger para presentar una documentación autogenerada, la cual se encuentra en la siguiente URL:

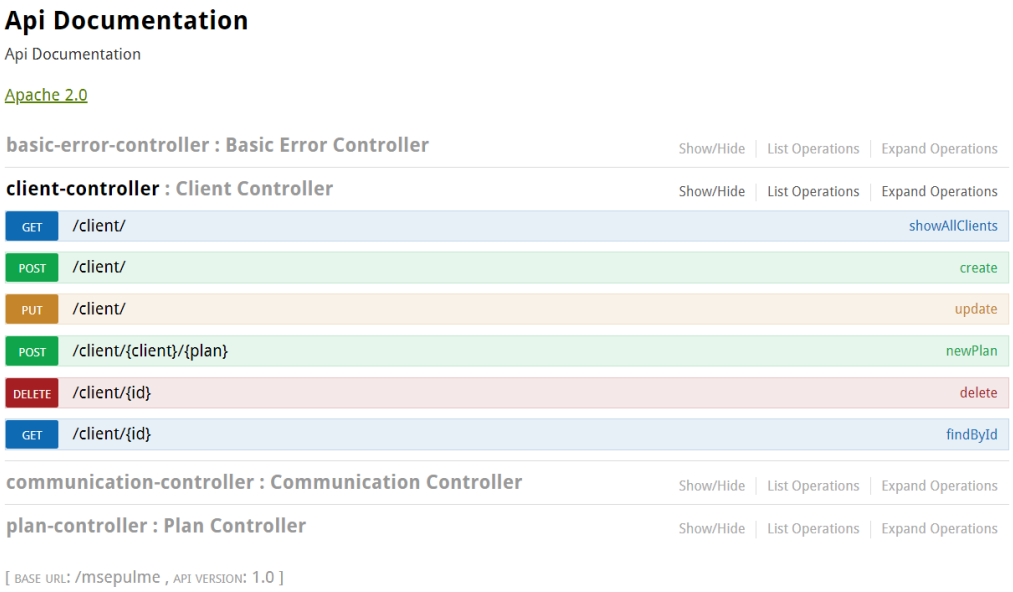
http://colvin.chillan.ubiobio.cl:8080/msepulme/swagger-ui.html

La cual muestra las diferentes clases controladoras como se presenta en la siguiente figura:



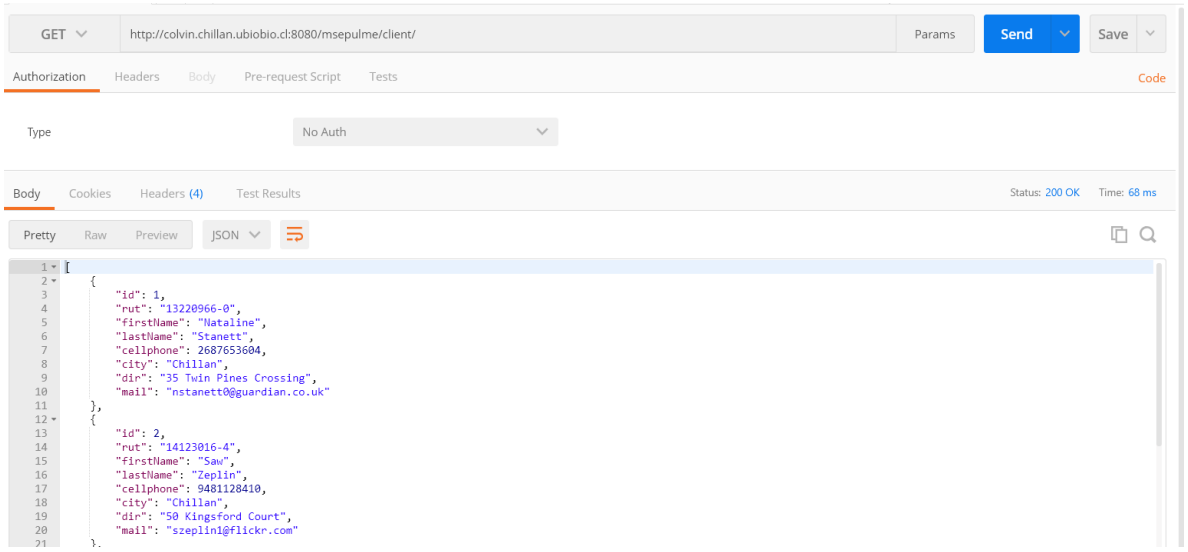
Figuras 19: Interfaz de Swagger

Y además, también muestra los detalles de los diferentes URIs, que provee cada una de estas clases, de forma que entrega el tipo de método HTTP soportado, la firma de la URI y la respuesta.



Figuras 20: Muestras URIs en Swagger

También puede usar un cliente como Postman para realizar los request a las URIs, como se presenta en la siguiente figura:



Figuras 21: Ejemplo con Postman

Para revisar cómo integrar Swagger a un proyecto revisar en los anexos, la sección 10.3

7 VALIDACIÓN DEL PRODUCTO

En esta sección se presentan las opiniones de profesionales inmersos en la industria del desarrollo de software y con experiencia en el uso de metodologías ágiles.

7.1 Opinión de Expertos

A los distintos expertos se les solicito responder las siguientes preguntas:

- ¿Según tu experiencia profesional, cree que este tipo de material este alineado con las metodologías de desarrollo actuales?

- ¿Crees que este tipo de material pueda apoyar a los estudiantes en su desempeño académico y luego profesional?

Aldemaro Díaz – UI Developer en Nisum Latam. – Chile

- ¿Según tu experiencia profesional, cree que este tipo de material este alineado con las metodologías de desarrollo actuales?

"El material presenta una buena introducción a la forma de generar productos de software en la actualidad. A través de este material los estudiantes podrán un primer acercamiento a las metodologías ágile"

- ¿Crees que este tipo de material pueda apoyar a los estudiantes en su desempeño académico y luego profesional?

"Si totalmente, académica y profesionalmente ya que les permite saber que hay metodologías para generar software de manera colaborativa, iterativa e incremental y de calidad."

Linkedin: https://www.linkedin.com/in/aldemarodiazh/

Henry Garcia – DevOps Engineer en Nisum Latam. – Chile

- ¿Según tu experiencia profesional, cree que este tipo de material este alineado con las metodologías de desarrollo actuales?

  "Si, ya que desarrolla temas usados en el campo laboral actual, donde las metodologías ágiles se usan para crear equipos altamente competitivos"

- ¿Crees que este tipo de material pueda apoyar a los estudiantes en su desempeño académico y luego profesional?

  "Si, porque los ejemplos reflejan la problemática que se presenta en el día a día en un equipo de desarrollo software."

Linkedin: https://www.linkedin.com/in/henry-omar-garc%C3%ADa-ortiz-9aaa5493/

8 CONCLUSIONES Y TRABAJO FUTURO

8.1 Conclusiones

La problemática que dio origen a este proyecto fue la dificultad que presentaron los estudiantes al cursar la asignatura de Ingeniería de Software, al desarrollar el "Portafolio de Proyecto de Software". Teniendo en cuenta estos comentarios, lo siguiente fue crear una propuesta para el desarrollo de un material de estudio que buscara apoyar el proceso de aprendizaje.

Para la gestión del desarrollo de este proyecto se decidió utilizar una modificación de la metodología de desarrollo tradicional Cascada a la que se le añadieron aspectos de Scrum, por lo que se consideraron cada una de las unidades del trabajo una historia de usuario.

Durante el desarrollo de este proyecto, se recibieron comentarios de diferentes partes, tanto como de docentes del DCCTI y grupos de estudiantes de la carrera Ingeniería Civil en Informática, de forma que fuera posible mantener un alcance apropiado del material de estudio presentado para cada una de las áreas cubiertas por este.

Lo que finalmente dio como resultado un material de estudio sencillo, que presenta de manera ordenada cada uno de los contenidos de forma que los conocimientos adquiridos en cada uno de ellos pueda tener un impacto en el desarrollo de la siguiente unidad, de forma que al llegar al último capítulo sea necesario aplicar todo lo visto anteriormente lo que implica entregarle a los estudiantes una guía de alto nivel para el desarrollo de un proyecto de software.

8.2 Trabajo a futuro

Al término de este trabajo queda pendiente la puesta en marcha del material de estudio en una prueba real con estudiantes que estén cursando activamente la asignatura de Ingeniería de Software y no solo se reciba feedback de estudiantes que ya hayan cursado esta, además se considera de vital importancia para el éxito de este trabajo los siguientes puntos:

- Realizar pruebas en un ambiente real en el que se pueda comprobar que el material de estudio presentado, puede generar cambios positivos en los resultados académicos de los estudiantes.

- Mantener el material actualizado a la última versión de Java y de Spring Boot según corresponda.
- Integrar nuevos contenidos y ejercicios propuestos, según las tecnologías, técnicas y metodologías evolucionan.
- Presentar otras técnicas de desarrollo basada en test, como el Behaviour-Driven Development.
- Considerar la integración de una contraparte para el desarrollo de un sitio utilizando tecnologías para el desarrollo Front End o implementando una aplicación para dispositivos móviles, e idealmente preparar un plan para capacitar a los estudiantes según sus intereses.
- Considerar la integración de una sección para explicar el impacto de la cultura DevOps en los proyectos de software.

9 REFERENCIAS

Agile Alliance. (s.f.). *What is Test Driven Development(TDD)?* Recuperado el 5 de Octubre de 2017, de https://www.agilealliance.org/glossary/tdd/

DCCTI. (s.f.). *Departamento de Ciencias de la Computación y Tecnología de la Información - Universidad del Bío - Bío*. Recuperado el 3 de Agosto de 2018, de http://www.ubiobio.cl/dccti/

Developer Mozilla. (s.f.). *REST – Glossary*. Recuperado el 3 de Marzo de 2018, de https://developer.mozilla.org/en-US/docs/Glossary/REST

Fowler, M. &. (1999). *Refactoring: improving the design of existing code.* Addison-Wesley Professional.

Gatica-Lara, F., & Uribarren-Berrueta, T. (2013). ¿Cómo elaborar una rúbrica? *Investigación en Educación Medica*, 61-65.

GitHub. (s.f.). *About*. Recuperado el 15 de Noviembre de 2017, de https://github.com/about

Jurado, C. B. (2010). *Diseño Ágil con TDD*. Obtenido de http://www.carlosble.com/downloads/disenoAgilConTdd_ebook.pdf

Ryan, F. (2017 de Junio de 2017). *Language Framework Popularity: A Look at Java*. Recuperado el 11 de Noviembre de 2017, de http://redmonk.com/fryan/2017/06/22/language-framework-popularity-a-look-at-java-june-2017/

Schwaber, K., & Sutherland, J. (2017). *The Scrum Guide*. Recuperado el 11 de Noviembre de 2017, de https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

TIOBE Index. (s.f.). *TIOBE Index*. Recuperado el 2018 de Junio de 18, de https://www.tiobe.com/tiobe-index/

Vicerrectoría Académica. (2008). *Modelo Educativo de la Universidad del Bío-Bío. Comisión de Renovación Curricular, Vicerrectoría Académica. 2008.* Concepción: Ediciones Universidad del Bío-Bío.

Walls, C. (2014). *Spring In Action 4th Edition.* Manning.

10 ANEXOS

En este apartado se exponen los resultados de la encuesta realizada a los estudiantes de la carrera de Ingeniería Civil en Informática, durante el periodo académico 2018-1.

10.1 Encuesta

Para identificar las necesidades de los estudiantes y poder determinar que era necesaria la construcción de un material de apoyo para la realización del "Portafolio de Proyecto de Software" se realizó una encuesta anónima con la finalidad de evaluar la necesidad de confeccionar dicho material.

La encuesta está compuesta por las siguientes preguntas:

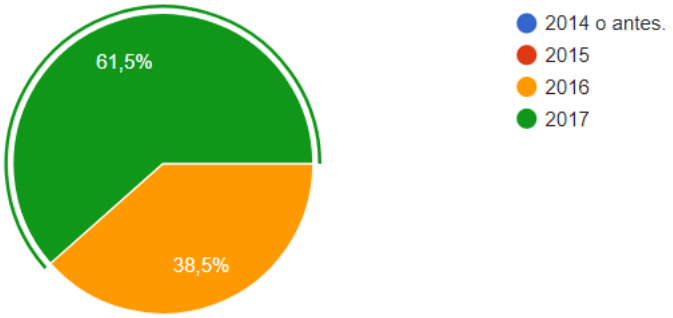
1. ¿En qué año realizaste la asignatura de Ingeniería de Software?
2. De 1, a 5, en donde 1 es bastante complejo y 5 con bastante facilidad, ¿Que tan sencillo fue realizar el proyecto de software que contempla la asignatura con los conocimientos que tenías hasta ese entonces?
3. Respecto al resultado de la pregunta anterior. ¿Cuál fue el principal detonante para calificar el desarrollo del proyecto del curso de esa manera?
4. Según tu experiencia, califica de 1 a 5, en donde 1, es insuficiente, y 5 es excelente.
   ¿El apoyo del docente fue suficiente en el proceso de la elaboración del proyecto semestral?
5. Evaluando según tu experiencia entre 1 y 5, en donde 1, es en desacuerdo y 5 es muy de acuerdo.
   ¿Crees que la existencia de un material de apoyo elaborado de forma previa hubiese hecho más sencillo el desarrollo del proyecto semestral?

10.2 Resultados Encuesta

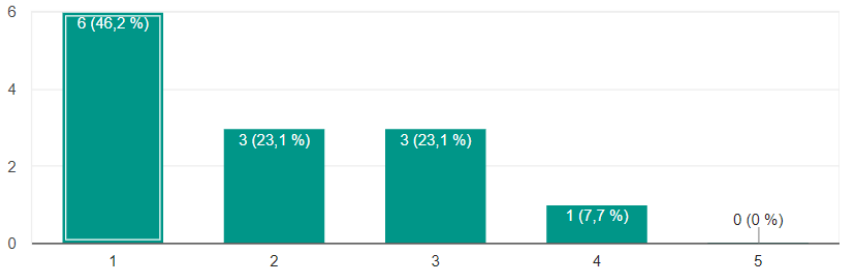
La encuesta fue realizada durante el segundo semestre del año 2017 y se repitió durante el primer semestre del 2018, en donde los estudiantes que respondieron la encuesta corresponden a una muestra de estudiantes que rindieron la asignatura durante los últimos años 4 años.

Dicha muestra corresponde a 13 estudiantes lo que corresponde a cerca de un 25% o 30% de los estudiantes que cursan la asignatura en un periodo académico, esta muestra fue obtenida mediante una encuesta anónima y voluntaria que se envió a los estudiantes en diversas ocasiones durante el los distintos periodos académicos.

¿En qué año realizaste la asignatura de Ingeniería de Software?



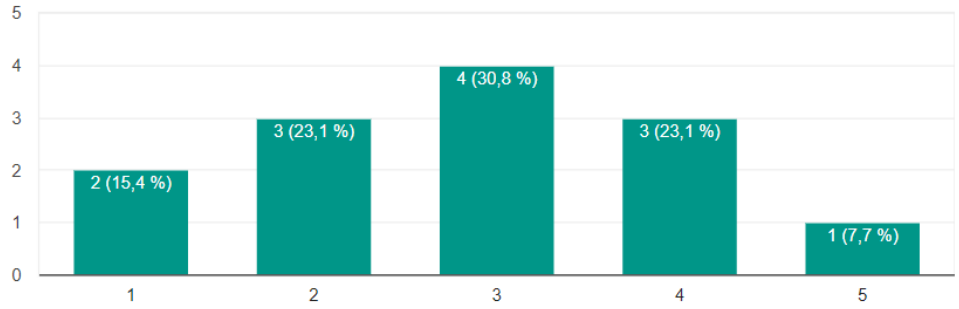
De 1, a 5, en donde 1 es bastante complejo y 5 con bastante facilidad, ¿Que tan sencillo fue realizar el proyecto de software que contempla la asignatura con los conocimientos que tenías hasta ese entonces?



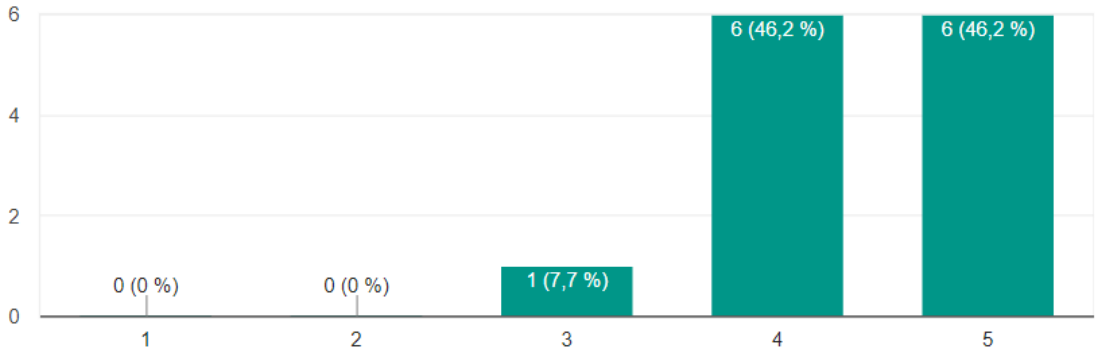
Respecto al resultado de la pregunta anterior. ¿Cuál fue el principal detonante para calificar el desarrollo del proyecto del curso de esa manera?

- Falta de estudio.
- La programación y conexión con la base de datos y el servicio web
- Baja motivación en las clases, plataformas absurdas que entorpecían el trabajo
- Problemas internos del equipo y el contraste de la teoría con la práctica
- El docente a cargo de guiar el proyecto entregó una plantilla que no explicó y no ayudó mayormente en el desarrollo
- Clases monótonas
- La falta de enseñanza de las tecnologías a utilizar.
- Porque el ayudante no ayudó y la poca experticia del grupo
- La complejidad de adaptar nuevas tecnologías con poco tiempo libre (considerando el contexto de desarrollar un proyecto durante el semestre con muchas otras asignaturas al mismo tiempo) y las exigencias específicas, por ejemplo, trabajar obligatoriamente con cierto IDE, lenguaje, etc. restringiendo mucho el desarrollo.
- Las tecnologías a usar, y falta de preparación para dicha tecnología.
- Difícil programar un proyecto grande

Según tu experiencia, califica de 1 a 5, en donde 1, es insuficiente, y 5 es excelente.
¿El apoyo del docente fue suficiente en el proceso de la elaboración del proyecto semestral?



Evaluando según tu experiencia entre 1 y 5, en donde 1, es en desacuerdo y 5 es muy de acuerdo. ¿Crees que la existencia de un material de apoyo elaborado de forma previa hubiese hecho más sencillo el desarrollo del proyecto semestral?



10.3 Definiciones de Tecnologías y Técnicas.

10.3.1 Test-Driven Development

El Test-Driven Development (TDD) se refiere a una técnica de programación en la cual se relacionan tres actividades diferentes: Codificación, Testing (en forma de Test Unitarios) y el Diseño (en la forma de Refactoring). (Agile Alliance)

El TDD puede ser descrito como la sucesión de las siguientes reglas:

- Escribe "un" solo test unitario que describa un aspecto del programa.
- Ejecuta el Test el cual debería fallar ya que el programa no cumple con esa funcionalidad.
- Escribe el código "necesario", de la manera más simple posible, para hacer que el test pase.
- "Refactoriza" el código hasta que cumpla los criterios de simplicidad.
- Y repite, "acumulando" test unitarios en el tiempo.

El TDD fue utilizado como la técnica de desarrollo para todo lo que está relacionado con la construcción de software dentro del material de estudio.

10.3.1.1 Criterios de simplicidad

Las reglas de simplicidad son una forma de las buenas prácticas de la programación que entrega diferentes reglas para evitar que el código se vea sucio o genere olores[17], las cuales son:

- El código debe ser verificado por test automatizados y además todos los test deben pasar.
- El código no debe tener duplicaciones.
- El código debe expresar de forma separada cada idea o responsabilidad.
- El código debe estar compuesto del mínimo de componentes (clases, métodos, líneas) que sea compatible con los tres primeros criterios.

[17] https://martinfowler.com/bliki/CodeSmell.html

10.3.2 Servicios Web del tipo REST

Los servicios web del tipo REST, hacen referencia al uso de la arquitectura API REST, la cual se provee una API pública la que hace uso de los métodos HTTP[18] para realizar sus operaciones. (Developer Mozilla)

Método HTTP	**Acción**
POST	Crear un recurso en el servidor
GET	Obtener un recurso del servidor
PUT	Actualizar un recurso del servidor
DELETE	Eliminar un recurso del servidor

Tabla 5: Métodos HTTP Soportados en REST.

10.3.3 Tecnologías

A continuación, se presentarán la tecnologías utilizadas durante el desarrollo del presente material de estudio.

- **Spring Framework**

Spring es un framework para el desarrollo de aplicaciones y contenedor para la inversión de control, de código abierto para la plataforma Java, el cual fue lanzado el año 2002 en su primera versión por Rod Johnson. (Walls, 2014)

- **API REST**

Representational State Transfer, o Transferencia de Estado Representacional, es un estilo de arquitectura para diseñar aplicaciones en red. Utiliza el protocolo HTTP, el cual permite compartir información entre cliente y servidor.

- **Structured Query Language(SQL)**

Es un lenguaje de programación diseñado para almacenar, manipular y recuperar datos almacenados en bases de datos relacionales.

[18] https://developer.mozilla.org/es/docs/Web/HTTP/Methods

- **H2**

Es un sistema administrador de bases de datos relacionales desarrollado en Java. Este puede ser incorporado en aplicaciones Java o ejecutarse de modo cliente-servidor.

En este caso, Spring contiene una base de datos H2 embebida, por lo cual se hará uso de está, ya que se está base de datos se pone en marcha en conjunto con la aplicación.

- **Java Development Kit 8**

Corresponde a la versión 1.8 de las librerías para el desarrollo de aplicaciones Java lanzada de manera oficial 18 de marzo del 2014, además considerado uno de los lenguajes más populares de la industria. (TIOBE Index)

- **Swagger**

Es una librería Open Source la cual permite autogenerar la documentación de una API REST desarrollada.

10.3.4 Herramientas

- **IntelliJ IDEA**

Es un entorno de desarrollo integrado para el desarrollo de aplicaciones informáticas producido por JetBrains.

- **GitHub**

Plataforma web para el control de versiones también conocido como una herramienta para la gestión de código fuente o repositorios.

- **Gradle**

Herramienta para automatizar la construcción del proyecto escrito en el lenguaje Groovy conocido como el sucesor de Maven.

- **H2-Web Console**

Consola para visualizar las tablas mediante una consola que funciona mediante el navegador.

- **Swagger-UI**

Permite integrar una documentación de todos los end-points disponibles en el sistema en el que es implementado.

- **Postman**

Aplicación de escritorio que permite acceder a un servicio REST mediante un entorno gráfico.

- **JUnit**

Librería para programar pruebas unitarias para el lenguaje de programación Java, fue lanzado el año 2012 y fue escrito por Kent Beck.

- **Mockito**

Framework de pruebas para el lenguaje Java el cual permite generar dobles de pruebas. Esto permite aislar los test a unidades específicas a la hora de escribir test unitarios.

10.4 Definiciones de Palabras Clave, Siglas y Abreviaciones

En esta sección se exponen los términos utilizados dentro del proyecto.

- **Backend:**

Es la parte de un software que contiene la lógica del negocio, y se encuentra constituida de forma que el usuario solo puede interactuar con esta, mediante una interfaz.

- **Cobertura de código:**

Corresponde a la cantidad de código que se encuentra cubierto por algún tipo de test, en este caso en particular, se considerarán solo test unitarios.

- **Code Review:**

Corresponde a la actividad que se realiza luego de terminar una funcionalidad, esta actividad busca reducir los errores del código y compartir conocimiento entre los desarrolladores.

- **Code Smell:**

Se le llama de esa manera, al código que posiblemente contenga algún problema, este nombre se le da a la acción de ver el código y "olfatearlo" inmediatamente.

- **Endpoint:**

Es el punto de conexión con la API Rest.

- **Frontend:**

Es la parte del Software que funciona como intermediario entre el cliente y el backend entregando una interfaz de usuario.

- **Historias de Usuario (HU):**

Es la especificación de requerimientos mediante un lenguaje natural y en pocas palabras, de forma que sea entendible y conciso.

- **Kanban:**

  Tablero para la organización de tareas, el cual organiza las tareas de manera que posee un listado de las tareas pendientes del trabajo, las tareas que se realizaran en la presente iteración, las tareas en ejecución, tareas en revisión y las tareas terminadas.

- **Mock:**

  Se le llama de esta manera a la instancia de una clase de pruebas la cual retornara datos predefinidos para enfocar las pruebas solo al SUT y no a las clases que interactúan con él.

- **Product Backlog:**

  Es el listado de historias de usuario que se deben desarrollar para dar por completado el proyecto.

- **Prueba Unitaria:**

  Corresponde a la unidad básica de pruebas automatizadas, la cual busca probar una y solo una funcionalidad de un método en específico.

- **Repositorio:**

  Es el espacio en la nube donde se guarda un proyecto.

- **Source Code Managment (SCM) – Sistema de Control de Versiones (SCV):**

  Es una forma de administrar los cambios en los documentos de un proyecto de Software.

- **Sizing:**

  Es la actividad que se realiza luego de tener el product backlog con todas las historias de usuario creadas, de esta forma se les asigna un tamaño a las tareas para poder tener un estimado del tamaño total del proyecto.

- **Sprint Backlog:**

  Corresponde a las historias de usuario que se desarrollarán en la presente iteración.

- **Subject Under Test (SUT):**

  Representa una clase que está siendo puesta a prueba por la batería de test unitarios.

- **Uniform Resource Locator (URL):**

Conocido en español como el Localizador de Recursos Uniforme, es la forma más común de administrar y encontrar información en la web.

- **Web Container:**

Para ejecutar una aplicación Java del lado del servidor, se requiere que alguien administre las peticiones al servidor y además se comunique con la JVM, en casos como este tenemos a Tomcat el cual tomará las peticiones y las pasara a la aplicación contenida dentro de él que corresponda.

10.5 Esquema Extenso – Ingeniería de Software

UNIVERSIDAD DEL BÍO-BÍO
La Libertad del Conocimiento

UNIVERSIDAD DEL BÍO-BÍO
VICERRECTORÍA ACADÉMICA – DIRECCION DE DOCENCIA

I. IDENTIFICACIÓN

<table>
<tr><td colspan="2">Nombre asignatura: INGENIERIA DE SOFTWARE</td><td rowspan="3">Período de Vigencia:
2016 - 2018</td></tr>
<tr><td colspan="2">Código: 634187</td></tr>
<tr><td colspan="2">Tipo de Curso: Obligatorio /Formación Disciplinar/ Ingeniería de Software/Semestral</td></tr>
<tr><td>Carrera: Ingeniería Civil en Informática</td><td>Departamento:
DCCTI y DSI</td><td>Facultad: Ciencias Empresariales</td></tr>
<tr><td>Nº Créditos SCT: 5</td><td>Total de horas:
Cronológicas: 162
Pedagógicas: 234</td><td>Año/ semestre: 4 / 8</td></tr>
<tr><td colspan="2">Horas presenciales: 108
HT:4
HP:2
HL:0</td><td colspan="2">Horas trabajo autónomo:126
HT:4
HP:3
HL:0</td></tr>
<tr><td colspan="2">Prerrequisitos:
Asignatura: SISTEMAS DE INFORMACIÓN
Código: 634184

Asignatura: BASE DE DATOS
Código:634180</td><td colspan="2">Correquisitos: Ninguno
Asignatura:
Código:</td></tr>
</table>

II.- DESCRIPCIÓN

II.1 Presentación: Relación de la Asignatura con las Competencias del Perfil de Egreso

Asignatura teórico práctica que al finalizar permitirá al estudiante construir software usando diversos elementos propios de la Ingeniería de Software. Se ubica en el octavo semestre, cuarto año, de la carrera. Contribuye a las competencias específicas:

CE.2.a.1 Diagnosticar la situación actual con el objeto de determinar los requerimientos de software de los usuarios.

CE.2.a.3 Construir aplicaciones de software, probando su funcionalidad y eficiencia, mediante el uso de arquitecturas, modelos, patrones, técnicas y herramientas de programación pertinentes para distintas plataformas.

Asimismo, contribuye a la competencia genérica:

CG.3 Establecer relaciones dialogantes para el intercambio de aportes constructivos con otras disciplinas y actúa éticamente en su profesión, trabajando de manera asociativa en la consecución de objetivos.

II.2 Descriptor de competencias (metas de la asignatura)

Construir software utilizando un conjunto de métodos, técnicas, herramientas y estándares que permitan obtener un producto que cumpla con los requerimientos técnicos, funcionales y no - funcionales.

1. Reconoce las características de los métodos de desarrollo de software, tradicional y moderno, para identificar las diferencias entre ellos.
2. Analiza la aplicabilidad de los métodos de desarrollo de software para identificar el más apropiado en un proyecto, considerando sus características y las de cada método.
3. Aplica de manera sistemática métodos, técnicas, herramientas y estándares de requerimientos, análisis y diseño para desarrollar un software que satisfaga los requerimientos técnicos, funcionales y no - funcionales.
4. Aplica técnicas de prueba de software para comprobar que un software satisface los requerimientos funcionales.

II.3 Aprendizajes Previos

Aplica técnicas de modelamiento de datos y funcionales.
Construye software utilizando lenguajes de programación.
Construye bases de datos que respondan a un conjunto establecido de necesidades.
Reconoce el contexto de negocios donde se insertan las aplicaciones de software.

III. RESULTADOS DE APRENDIZAJE

Resultados de Aprendizaje	**Metodología**	**Criterios de Evaluación**	**Contenidos conceptuales, procedimentales y actitudinales.**	**Tiempo estimado**
Reconoce las características de los métodos de desarrollo de software tradicionales y modernos para identificar las diferencias entre ellos.	Método expositivo con discusión socializada. Trabajo individual y colaborativo.	1. **Describe** el concepto de proceso de desarrollo de software explicando sus características y las actividades/áreas que lo conforman. 2. **Describe** el concepto de producto software explicando sus componentes. 3. **Describe** el concepto de modelos de calidad explicando su propósito y	**Conceptual** Proceso de desarrollo de software, producto y calidad Métodos de desarrollo de software tradicionales y modernos (secuenciales, incremental, iterativos, ágiles), características, fases y/o actividades. **Procedimental** Identificación de las diferencias entre los métodos de desarrollo de	Horas presenciales: HT: 28 HP: 14 HL: Horas de trabajo autónomo: HT: 28 HP: 21 HL:

Resultados de Aprendizaje	Metodología	Criterios de Evaluación	Contenidos conceptuales, procedimentales y actitudinales.	Tiempo estimado
		características. 4. **Describe** las características de los métodos de desarrollo de Sw., tradicionales y modernos, explicando sus características.	software.	
Analiza la aplicabilidad de los métodos de desarrollo de software para identificar el más apropiado en un proyecto, considerando sus características y las de cada método.	Método expositivo con discusión socializada. Trabajo individual y colaborativo Análisis de casos	5. **Identifica** fortalezas y debilidades de los métodos de desarrollo de Sw. en función del contexto y riesgos de un proyecto (aspectos relacionados con el personal, producto y proceso). 6. **Selecciona** el método de desarrollo de sw más apropiado para un proyecto basado en las fortalezas y debilidades del método y del contexto y riesgos de un proyecto. 7. **Explica** con argumentos claros las razones que justifican el método seleccionado para un proyecto dado.	**Conceptual** Características de un proyecto de desarrollo de software que inciden en el la selección de un método de desarrollo. **Procedimental** Análisis de aplicabilidad de un método frente a un proyecto de desarrollo de software. Justificación del método más apropiado. **Actitudinal** Pensamiento Crítico en el análisis de los métodos de desarrollo de software.	Horas presenciales: HT: 12 HP: 6 HL: Horas de trabajo autónomo: HT: 12 HP: 9 HL:
Aplica de manera sistemática métodos, técnicas, herramientas y estándares de requerimientos,	Método expositivo con demostraciones y tipos de pregunta Trabajo colaboraivo	8. **Describe** el proceso de gestión de requerimientos y algunos métodos y técnicas de ingeniería de requerimientos,	**Conceptual** Ingeniería de requisitos. Análisis del Sw. Diseño del Sw (funcional, datos y arquitectura). **Procedimental**	Horas presenciales: HT: 20 HP: 10 HL: Horas de trabajo autónomo:

Resultados de Aprendizaje	Metodología	Criterios de Evaluación	Contenidos conceptuales, procedimentales y actitudinales.	Tiempo estimado
análisis y diseño para desarrollar un software que satisfaga los requerimientos técnicos, funcionales y no - funcionales.	(rompecabeza y escucha enfocada) Proyecto,	tales como casos de uso, entrevistas, cuestionarios, entre otros. 9. **Emplea** algunos métodos y técnicas de ingeniería de requerimientos, tales como casos de uso, entrevistas, cuestionarios, entre otros. 10. **Utiliza** un estándar de documentación de de producto software en cada etapa del desarrollo. 11. **Utiliza** un método, técnicas de análisis y diseño en el desarrollo de un software.	Método de desarrollo de Sw. Técnicas de requerimientos, análisis y diseño. Uso de herramientas CASE. Uso de un estándar de documentación de producto software. **Actitudinal** Rigurosidad en el uso de métodos, técnicas, herramientas y estándares de Requerimientos, Análisis y Diseño en el desarrollo de un software.	HT: 20 HP: 15 HL:
Aplica técnicas de prueba de software para comprobar que un software satisface los requerimientos funcionales.	Método expositivo con demostraciones y tipos de pregunta Trabajo grupal	12. **Describe** el proceso y niveles de prueba de software explicando las actividades y propósito de cada nivel. 13. **Describe** los enfoques y técnicas para definir datos y casos de prueba, tales como partición equivalente, valor límite, caminos de prueba, entre otros. 14. **Utiliza** las técnicas para definir datos y casos de prueba a nivel de unidad,	**Conceptual** Proceso y niveles de prueba de software. Enfoques y técnicas de Prueba de software. **Procedimental** Utilización de técnicas de prueba en la definición de datos y casos de prueba. Ejecución de los casos de pruebas en el software.	Horas presenciales: HT:12 HP:6 HL: Horas de trabajo autónomo: HT: 12 HP: 9 HL:

Resultados de Aprendizaje	Metodología	Criterios de Evaluación	Contenidos conceptuales, procedimentales y actitudinales.	Tiempo estimado
		sistema y aceptación, tales como partición equivalente, valor límite, caminos de prueba, entre otros. 15. **Ejecuta** los casos de pruebas en el software para comprobar que funciona correctamente y satisface los requerimientos funcionales.		

IV. SISTEMA DE EVALUACIÓN

RESULTADOS DE APRENDIZAJE	EVIDENCIAS DE APRENDIZAJE (proceso y producto)
Reconoce las características de los métodos de desarrollo de software, tradicionales y modernos, para identificar las diferencias entre ellos.	Informe Trabajo Práctico Individual Informe Trabajo Grupal Control de Lectura
Analiza la aplicabilidad de los métodos de desarrollo de software para identificar el más apropiado en un proyecto, considerando sus características y las de cada método.	Informe Trabajo Práctico Individual Informe Trabajo Grupal Exposición Trabajo Práctico Grupal Certamen I (acumulativo de RA 1 y RA 2)
Aplica de manera sistemática métodos, técnicas, herramientas y estándares de requerimientos, análisis y diseño para desarrollar un software que satisfaga los requerimientos técnicos, funcionales y no - funcionales.	Portafolio del Proyecto de desarrollo de software. Control de Lectura
Aplica técnicas de prueba de software para comprobar que un software satisface los requerimientos funcionales.	Informe Trabajo Práctico Individual Informe Trabajo Grupal Certamen II (acumulativo de RA 3 y RA 4)

	(%)
La evaluación de la asignatura considera:	
CERTAMEN I y II	50
PORTAFOLIO DEL PROYECTO DE DESARROLLO DE SOFTWARE.,	20
CONTROL DE LECTURA	10
INFORME TRABAJO PRÁCTICO, (grupal e individual).	20

V. BIBLIOGRAFÍA

Fundamental

- BOHEM, B. 2006; A View of 20th and 21st Century Software Engineering. The International Conference on Software Engineering (ICSE). Shanghai, China.
- PRESSMAN, R. 2005. Ingeniería del Software, un enfoque práctico. Editorial McGraw Hill. 6ta edición.
- SOMMERVILLE, I. 2002. Ingeniería de software. Editorial Pearson Educación, 6ta edición.
- SHARI, P. 2002; Ingeniería de Software: Teoría y Práctica. Pearson Education.

Complementaria

- BRAUDE, E., Ingeniería de Software. Una Perspectiva Orientada a Objetos. Alfaomega. 2003.
- LARMAN, C. UML y Patrones. Una Introducción al Análisis y Diseño Orientado a Objetos y al Proceso Unificado, 2da. edición. Pearson. 2003.
- RUMBAUGH, BOOCH y JACOBSON, Lenguaje Unificado de Modelado. Pearson. 2007
- STEVENS, P., Utilización de UML en Ingeniería del Software con Objetos y Componentes, Pearson, 2007.
- SOFTWARE ENGINEERING INSTITUTE (SEI), Carnegie Mellon University (hhtp://www.sei.cmu.edu/)
- THE OBJECT MANAGEMENT GROUP (OMG) (http://www.omg.org/)
- WYSOCKI, Robert K. 2006. Effective Software Project Management. ISBN: 978-0-7645-9636-0. Wiley Publishing Inc. 1° Edición.

10.6 Integración de Swagger en el prototipo.

Dentro de este proyecto se utiliza Swagger para la auto-documentación de las URIs, por lo que para utilizarlo, se deben añadir las dependencias correspondientes al archivo ***build.gradle*** como se muestra en la siguiente figura:

```
dependencies {
    compile('org.springframework.boot:spring-boot-starter-data-jpa')
    compile('org.springframework.boot:spring-boot-starter-web')
    compile("org.springframework.boot:spring-boot-devtools")
    compile('org.springframework.boot:spring-boot-starter-tomcat')
    runtime('com.h2database:h2')

    compile group: 'io.springfox', name: 'springfox-swagger2', version: '2.6.1'
    compile group: 'io.springfox', name: 'springfox-swagger-ui', version: '2.6.1'

    testCompile('org.springframework.boot:spring-boot-starter-test')
}
```

Figuras 22: Añadiendo Swagger

Para luego poder implementar la siguiente clase:

```
@Configuration
@EnableSwagger2
public class SwaggerConfiguration {

    private static final Logger LOG = Logger.getLogger(SwaggerConfiguration.class.getName());

    @Bean
    public Docket api() {
        LOG.info( msg: "Swagger initialization...");
        return new Docket(DocumentationType.SWAGGER_2).select().apis(
                RequestHandlerSelectors.any()).paths(PathSelectors.any()).build();
    }
}
```

Figuras 23: Implementación Swagger

En esta clase además se añade un Logger, para que este pueda mostrar información mediante la consola.