Sistema Cafetería Virtual UNIAJC

Nicolás Bermúdez Rodríguez,✉ nocicu97@gmail.com
Nicolás Ortiz Acevedo, ✉ nico.ortiz.a@gmail.com

Proyecto presentado para optar al título de Ingeniero en Sistemas

Director: Manuel Alejandro Pastrana Pardo, Magister (MSc) Ingeniería de Software

Institución Universitaria Antonio José Camacho
Facultad de Ingenierías
Ingeniería de sistemas
Cali - Colombia
2020

Nota de aceptación:

Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Institución Universitaria Antonio José Camacho para optar al título de Ingeniero de Sistemas

Jurado

Jurado

Santiago de Cali, 8 Marzo de 2020

Dedicatoria

A mi padre Carlos Alberto Bermúdez González, que a pesar que hace unos años no puede estar conmigo, siempre lo llevo en mi espíritu, siguiendo los pasos que me enseño, dándome fortaleza y ánimos para nunca rendirme y dar mi mayor esfuerzo para lograr mi sueño de ser profesional.

A mi madre Sonia Rodríguez Quintero que es mi mayor motivación, siendo la persona que me apoyo desde el inicio de este gran proceso, siempre dándome consejos y haciendo sacar lo mejor de mí, logrando que sea una persona súper dedicada y comprometida en mis metas. Gracias por siempre confiar en mí.

Agradecimientos

Agradezco a Dios por guiarme y darme la fuerza de elegir el camino correcto para superar los momentos de dificultad, los momentos de duda y permitirme estar por culminar uno de los procesos más importantes de mi vida

Agradecemos también a los Ingenieros: Ana Milena Rojas y Manuel Pastrana porque creyeron en nuestras capacidades, porque siempre nos guiaron en un camino constante de crecimiento tanto como persona, como profesional

A mi compañero de tesis le agradezco por todo él es esfuerzo y empeño que los dos le hemos puesto para ser mejores como seres humanos y como futuros ingenieros en sistemas, a pesar de todas las adversidades que pudieron salir en el proceso de formación como ingenieros de la UNIAJC

A las personas que siempre nos dieron ánimo, colaboración y jamás dudaron de nuestras capacidades, hermanas, pareja sentimental, compañeros y amigos

Contenido

Resumen vii
Abstract viii
Introducción ix
Capítulo I 1
1.1 Problema de Investigación 1
1.1.1 Descripción del problema 1
1.1.2 Planteamiento 2
1.1.3 Preguntas de Investigación 3
Objetivos 3
1.2 Objetivo general 3
1.2.1 Objetivos específicos 3
1.3 Justificación 4
1.4 Alcance 4
1.5 Limitaciones 8
Capítulo II 8
2.1 Marco Referencial 8
2.1.1 Marco Contextual 8
2.1.2 Marco Teórico 10
2.1.3 Marco Conceptual 11
2.1.4 Antecedentes 12
Capítulo III 13
3.1 Metodología 13
Capitulo IV 15
4.1 Desarrollo del proyecto 16
4.1.1 Análisis 16
4.2 Planeación 21
4.2.1 Sprint Planning 21

4.3 Diseño 23

4.3.2. Arquitectura 28

4.3.2.1 Vista Física 28

4.3.2.2 Vista de Desarrollo 29

4.4 Construcción 30

4.4.1 Desarrollo de Aplicación Móvil 34

4.4.2 Definición de Tecnologías 37

4.5 Pruebas 38

5. Capítulo V 46

5.2. Conclusiones 46

5.3. Recomendaciones y trabajos futuros 47

6. Referencias 48

Lista de tablas

Tabla 1 Riesgos y estrategia identificadas 19

Tabla 2 Formato Historia de Usuario Epica, Basado en (Cohn, 2004) 20

Tabla 3 Atributos de calidad 24

Tabla 4 Atributos de calidad no observables 25

Tabla 5 Comparación entre servicios SOAP y servicios Rest. Fuente propia 32

Tabla 6 Descripción de las carpetas/archivos API REST. Fuente propia 33

Tabla 7 Descripción de las carpetas/archivos Ionic. Fuente propia 34

Tabla 8 Descripción de notificaciones push 37

Tabla 9 Descripción de tecnologías App Ionic. Fuente propia 37

Tabla 10 Descripción de tecnologías API. Fuente propia 37

Tabla 11 Descripción tabla estadísticas SoapUI, Fuente Documentación oficial SoapUI 39

Lista de ilustraciones

Ilustración 1 Arquitectura MVVM Aplicación Comunicación Académica. Fuente propia 27
Ilustración 2 Vista física (diagrama de despliegue). Fuente propia 28
Ilustración 3 Vista de desarrollo (diagrama de componentes). Fuente propia 29
Ilustración 4 Diseño web service Fuente propia 31
Ilustración 5 Diseño carpetas Ionic. Fuente propia 34
Ilustración 6 Tablero de pruebas de carga SoapUI, resultados de las pruebas del API en SoapUI, Fuente propia 39
Ilustración 7 Prueba automatización Katalon, Fuente propia 41
Ilustración 8 vista general de ambiente de calidad Fuente propia 42
Ilustración 9 Configuración de herramientas aseguramiento de calidad, Fuente propia 42
Ilustración 10 Dashboard Sonar Métricas Calidad (Estado Inicial), Fuente propia 43
Ilustración 11 Dashboard Sonar Métricas Calidad (Revisión 2), Fuente propia 43
Ilustración 12 Dashboard Sonar Métricas Calidad (Revisión 2), Fuente propia 44
Ilustración 13 Dashboard Sonar Métricas Calidad (Revisión 3), Fuente propia 44
Ilustración 14 Dashboard Sonar Métricas Calidad (Revisión final), Fuente propia 45
Ilustración 15 Dashboard Sonar Métricas Calidad (Revisión final), Fuente propia 45

Resumen

La institución universitaria Antonio José Camacho (UNIAJC), a través del proyecto de investigación denominado Smart Campus que busca un mejoramiento de los diversos servicios ofertados para la comunidad académica (estudiantes, docentes, administrativos y egresados) y que permite mejorar plenamente el funcionamiento en cada una de las respectivas áreas de la institución. Smart Campus como proyecto cuenta con unos ejes de trabajo en el que destaca Smart Living. Este eje impulsa un estilo de vida saludable, mejorando hábitos mediante recomendaciones prácticas y continuas. Una de las problemáticas que se identificaron que afectan a la comunidad son las excesivas colas en las horas pico evidenciando como el limitado tiempo que poseen no es suficiente para consumir los alimentos de la cafetería. Este trabajo de grado, busca alinear esfuerzos con el macro proyecto Smart campus, dentro del eje mencionado permitiendo ofertar un servicio de cafetería virtual, destacando que además de las compras por medio de la aplicación deben recogerse en la cafetería, se pueden realizar la reserva de almuerzos, visualización de todos los productos ofrecidos por la cafetería de la institución y la calificación del servicio ofrecido por la cafetería. La app está desarrollada bajo tecnologías multiplataforma por lo que estará disponible en sistemas operativos IOS y Android. El desarrollo del software fue realizado utilizando el proceso estándar de Smart campus basado un marco de trabajo ágil compuesto por los Scrum y XP.

Palabras clave: Smart Campus, Smart Living, API Rest, Cafetería UNIAJC, Android, iOS.

Abstract

In the last few years, the percentage of population with smartphones has been increasing significantly. The use of the mobile devices has become part of the normal life. The Antonio José Camacho Institution (UNIAJC) aware of this, is developing a project of investigation named Smart Campus which search the improvement the services offer to its community. Smart campus is composed of several axes, one of them called Smart Living. This one promotes a healthy community lifestyle, improving their habits through practical and continuous recommendations. One of the habits that affect the community is the consumption of food. For this reason, this degree work aligning efforts with smart campus in the mentioned axis, wants to build for the UNIAJC a virtual cafeteria service, which offers services such as purchases through the application, the reservation of lunches, the display of all products offered by the cafeteria of the institution and the qualification of the service offered by the cafeteria. The app, is developed under multiplatform technologies, so it will be available in IOS and Android. The software development process was made using the Smart Campus process based on an agile framework like Scrum and XP.

Keywords: Smart Campus, Smart Living, API Rest, Cafeteria UNIAJC, Android, iOS

Comentado [A1]: Coffee shop es donde venden donde vende cafésolo Las cafeterías se llaman en ingles cafetería

Introducción

El proyecto Smart Campus inscrito en la Facultad de Ingeniería de la Institución Universitaria Antonio José Camacho liderado por la Ingeniera Ana Milena Rojas del grupo de investigación GrinTIC con colaboración del semillero IT media, está desarrollando diversas soluciones orientadas al mejoramiento de la institución. Esta labor, incluye la colaboración activa de estudiantes en búsqueda del potenciamiento de su formación académica y su perfil profesional, incrementando sus habilidades en diversas áreas de la ingeniera de sistemas.

Smart campus cuenta con unas subcategorías específicas o ejes donde desarrolla los proyectos relacionados con el mejoramiento institucional, estos son:

- **Smart Living:** Desarrolla servicios que fomenten la integración social, la inclusión, el bienestar, la cultura y la empleabilidad
- **Smart Goverment**: Busca la participación de toda la comunidad y la transparencia de la política y la toma de decisiones.
- **Smart Enviroment**: Centra sus esfuerzos en el uso de las TI para optimizar, gestionar y racionalizar correctamente los recursos naturales.
- **Smart People**: Se enfoca en las habilidades, trabajo y teletrabajo mediante las TI, disponer de acceso a la educación, a la enseñanza y aprendizaje continuo tanto a estudiantes como a personal de la propia universidad, sociedad inclusiva que mejore la creatividad, la innovación y la participación en la vida pública.
- **Smart Economy**: Se enfoca en la educación como recurso base de la economía de la universidad, fomentando la competitividad e impulsar el emprendimiento personal, profesional e innovador
- **Smart Mobility**: Tiene como objetivo gestionar de manera sostenible, segura y eficiente la accesibilidad y los transportes de la universidad.

Alienando esfuerzos con el proyecto mencionado, específicamente dentro del eje Smart living, se concibe este trabajo de grado llamado “Sistema Cafetería Virtual UNIAJC”, que mediante un aplicativo permite ofrecer el servicio de las cafeterías de una forma más accesible.

Las cafeterías de la universidad, ofrecen sus productos para el consumo de la comunidad universitaria y en su día a día se ven afectados por los diferentes horarios que tienen los estudiantes, docentes, directivos en la hora de almuerzo haciendo que sus productos no tengan una demanda establecida. Este proceso afecta a la comunidad, porque en algunas ocasiones no pueden obtener el servicio deseado. Por lo anterior, surge la posibilidad de realizar el presente trabajo de grado con el fin de generar una alternativa para esta problemática, donde a través de una app, se pueda realizar pedidos como estudiantes, profesores y administrativos de la Institución Universitaria. Esto optimizará tiempos de espera y facilita la comprar sin necesidad de hacer filas en las cafeterías.

Para el desarrollo del este aplicativo se utiliza la metodología de desarrollo de software de Smart campus compuesto por los frameworks Scrum y XP. Esta metodología, buscando un mejor resultado de cara al desarrollo del aplicativo, mantiene como eje transversal de todo el proceso la calidad preventiva. La construcción del aplicativo será bajo el lenguaje de programación IONIC 3, permitiendo un desarrollo hibrido para las plataformas IOS y ANDROID, logrando alcanzar así a toda la comunidad.

Capítulo I

1.1 Problema de Investigación

1.1.1 Descripción del problema

La Institución Universitaria Antonio José Camacho, es una Institución de educación superior que en los últimos 10 años ha presentado un crecimiento de 3.166 a 7.500 estudiantes basado en Fernando Cifuentes(2018). Cuenta con 4 facultades, con un total de 38 programas en oferta académica para los diferentes niveles de formación en modalidad presencial y semipresencial.

No obstante, el crecimiento que ha experimentado, se ve afectado por el número de estudiantes, llevando a la institución a tener que reorganizar algunos de sus procesos y servicios para brindar una atención de alta calidad. Uno de los servicios mencionados es la cafetería de la institución que actualmente hace un buen trabajo con los recursos disponibles, pero puede generar ciertos problemas en horas de incremento de atención e incluso presentar escases de productos para el consumo por la alta demanda en algunas horas del día. Esto sucede porque actualmente este servicio no cuenta con una herramienta de control y seguimiento que le permita prever esta situación.

Lo anterior refleja que la comunidad universitaria no dispone de herramientas que le permita prevenir estas situaciones, dificultando el proceso natural de compra y casi obligando a buscar otras alternativas por afuera de la institución. Con una herramienta que permita facilitar este proceso, se minimizarían los contratiempos a la hora de consultar la disponibilidad de los productos y las cafeterías no tendrían problemas de reabastecimiento si tiene alertas tempranas, causando un impacto positivo en este servicio.

1.1.2 Planteamiento

Las cafeterías pertenecientes a la Institución Universitaria Antonio José Camacho, son las encargadas de brindar el servicio de alimentación a la comunidad dentro del plantel. Con el propósito de mejorar su servicio y tener una mayor competitividad con los restaurantes cercanos a la institución, se plantearon unas posibles estrategias para tener una mayor eficiencia en su servicio buscando atraer a toda la comunidad perteneciente a la institución. Además de ello, existe una oportunidad de mejorar el servicio de la cafetería por la ineficiencia que tiene al momento de manejar las horas pico, pues en ese momento, es donde más presentan escasez de productos.

A raíz de esto, surgen varios problemas importantes. Uno de ellos es ¿Cómo se puede realizar una mejora del servicio para que toda la comunidad de la institución pueda tener acceso a la visualización de los productos de la cafetería desde cualquier lugar en que se encuentre? Otra problemática, que afecta directamente a la cafetería es saber ¿cuántas personas desean algún producto? Para así manejar eficientemente el inventario de productos sin generar desperdicios o escases.

Desde Smart Campus, se propone como estrategia de mejoramiento presentando los servicios de la cafetería desde los dispositivos móviles con una aplicación, permitiendo que cualquier persona perteneciente a la institución pueda en todo momento verificar que productos están disponibles en la cafetería, que almuerzos tiene ofertados para esta semana y poder realizar el proceso de reserva y compra de su alimento a través de su dispositivo móvil de confianza. Además, esto optimizaría tanto la venta de productos como el manejo de inventario de las cafeterías teniendo un impacto ecológico en la no generación de desechos innecesarios por alimentos que no se vendan.

1.1.3 Preguntas de Investigación

- ¿Cómo elicitar correctamente las necesidades funcionales de las cafeterías y de la comunidad para crear una aplicación capaz de satisfacerlas?
- ¿Cómo diseñar una solución correcta basada en los atributos de calidad observables y no observables que guían la arquitectura de software del sistema?
- ¿Cuáles son las tecnologías correctas involucradas en el desarrollo de la solución?
- ¿Qué mecanismos de calidad preventiva y correctiva será utilizados para medir la solución?

Objetivos

1.2 Objetivo general

Desarrollar un sistema de información móvil multiplataforma que facilite el uso de los servicios ofrecidos por la cafetería a la comunidad universitaria.

1.2.1 Objetivos específicos

- Realizar la elicitación de requisitos mediante una técnica que permita unificar la visión del proyecto tomando en la cuenta la opinión de todos los interesados de la comunidad.
- Generar una arquitectura de software lo suficientemente eficaz para cumplir con las necesidades expuestas por los atributos de calidad de la solución.
- Desarrollar la solución siguiendo el modelo de desarrollo de software de Smart campus.
- Garantizar la calidad y mejora continua del aplicativo mediante el ambiente DevOps ofertado por el proyecto Smart Campus.

1.3 Justificación

El proyecto está enmarcado dentro del plan estratégico de desarrollo conforme a los lineamientos de un objetivo estratégico de la universidad, todo el proyecto se encuentra enmarcado en los ámbitos de desarrollo del ecosistema Smart campus el cual corresponde a modernizar la infraestructura organizacional de los procesos académicos de gestión a través de un modelo universidad inteligente soportada en prestar servicios y recursos digitales, buscando acercar los servicios virtuales a través de plataforma digitales para que los usuarios tengan una mayor interacción con estos buscando una mejor calidad de vida, que la cafetería brinde el servicio donde se desee, en el momento que se requiera para resolver necesidades de la comunidad académica.

Para ello se procede a contrastar esta información con el personal de cafetería y con la comunidad universitaria obteniendo una respuesta bastante positiva que da viabilidad del proyecto, como se llegó a esta conclusión, fue por medio de una encuesta realizada como anterior mente se mencionan los estudiantes de la UNIAJC (Anexo Encuesta interés) con la encuesta realizada se identifica que el proyecto es muy pertinente.

1.4 Alcance

Para la elaboración del aplicativo se utilizó la metodología de desarrollo de software de Smart Campus que combina lo mejor de los frameworks SCRUM y XP, y que está basada en el trabajo de Pastrana, Ordóñez, Rojas, & Ordoñez (2019). Esta metodología utiliza las etapas del ciclo de desarrollo de software descritas por Pressman(2012) a continuación:

- **Análisis y Planeación:** En esta etapa el principal objetivo es determinar las necesidades del producto a realizar, se pueden utilizar varias herramientas técnicas

para elicitar la información para dicho proyecto. Smart campus utiliza la técnica Inception Deck para recopilar la información necesaria, esta técnica según Pastrana, Ordóñez, Ordóñez, & Merchan (2017) consiste que tiene tableros cada uno de ellos tiene un propósito los cuales son:

- ¿Quién está en el auditorio? Permite saber quién está en la reunión.
- ¿Por qué estamos aquí? Esta pregunta va enfocada a la visión del producto.
- Elevator pitch. Consiste en identificar rápida mente cual es la esencia del proyecto.
- Diseñar la caja del producto. Identifican 4 cosas esenciales las cuales son nombre del producto, frase que lo identifica, beneficios del producto y que lo diferencia de los demás.
- Lista del no corresponde a la delimitación del producto.
- Conoce el vecindario. Permite identificar como las personas interactúan para creas relaciones y permita un éxito del proyecto.
- ¿Qué te impide dormir en las noches? Identifica de forma temprana riegos que puede tener el proyecto y las estrategias a seguir para evitar que sucedan o si suceden que su impacto sea el mínimo posible
- Mostrar la solución. Permite delimitar funcionalmente el proyecto
- ¿Lo que hay que dar? Que todas las personas que están involucradas tengan claro que compromisos deben asumir para el éxito del proyecto.
- ¿Lo que costara hacerlo? No en términos económicos si no en aspectos técnicos y riesgos.

Al finalizar la técnica, la información recopilada en se transcribe en historias de usuario para conformar un product backlog. Este artefacto, sirve de base para la planificación.

Bajo la forma de trabajo de Smart campus, la estimación se realiza mediante la técnica de póker scrum. Esta cuenta con una baraja de cartas cada una con un peso (numérico) que simboliza que tan difícil es realizar una historia de usuario pensado desde varias perspectivas (analizar la información dada, diseñar una

solución, desarrollarla, probarla, ajustarla si es necesario, etc). Por parte de la priorización, Smart campus utiliza la técnica moscow. Esta ayuda a definir el orden de las necesidades y en qué momento se debe realizar. los eventos de scrum utilizados para mantener a todos informados del avance del proyecto y de la gestión de riesgos son Planning meeting y daily scrum.

- **Diseño:** En esta etapa es fundamental la consolidación de los atributos de calidad y los patrones de diseño que guían el estilo arquitectural de la solución. El artefacto resultante de esta etapa es el Documento de Arquitectura de Software o DAS que contiene las decisiones que guía la construcción del aplicativo y son plasmadas mediante diagramas UML. El modelo base utilizado para saber que diagramas UML deben reflejar la solución es fundamentado en el trabajo de Kruchten (1995). Cabe aclarar que al seguir una metodología basada en frameworks agiles, no se modelan todas las vistas, sino solamente las necesarias para reflejar correctamente la solución.

- **Desarrollo:** Esta etapa se generan los eventos daily srum, sprint review, sprint retrospective, y por último el sprint el cual es donde se realiza la entrega de la historia de usuario completamente funcional, además de ello se da uso del ambiente de Devops el cual nos permite Smart campus, una de las herramientas más importantes es sonarQube el cual ayuda a identificar las malas prácticas y con esto desde un inicio el código será más robusto y no presente bugs cuando esté en su versión final.

Calidad o pruebas: A diferencia de las metodologías de desarrollo tradicional, las que están basadas en frameworks agiles no buscan que la calidad sea correctiva sino preventiva. Aquí el eje central es trabajar la calidad al tiempo que se avanza en el desarrollo del proyecto. Al ser evolucionado de manera iterativa incremental, las posibilidades de mejora del entregable son inmediatas, facilitando un crecimiento sostenible y con el mínimo de reproceso. Las buenas practicas

sugeridas por Pastrana, Ordóñez, Rojas, & Ordoñez (2019) son implementadas en su totalidad.

Despliegue: En el proyecto está construido teniendo en cuenta una división de frontend y backend. Dentro del back, se construyen servicios tipo REST que exponen la lógica de negocios y que serán consumidos por la aplicación móvil que contiene el frontend o pantallas del sistema.

Para lograr esta comunicación, al equipo de Smart campus se le entregan el script de BD Postgres y un archivo de empaquetado war que contiene la lógica mencionada anteriormente. Esto se expone a través del servidor de Smart campus con la dirección URL: (https://smartdev.uniajc.edu.co:8444/) la cual después de que el equipo de Smart campus despliega el war se pueden realizar peticiones al servicio, la cual realiza la conexión a la base de datos de Posgrest y a el login de Smart campus, luego de ello se realiza es generar el apk el cual realizara las peticiones a dicho servicio.

Complementando lo anterior y dando más detalle sobre el aplicativo los módulos que compondrán el sistema son:

- **Módulo de autenticación:** Este módulo permite el consumo del servicio web de autenticación a través de usuarios de academusoft, creado por Smart campus. Aquí el sistema realiza una integración con esta plataforma para garantizar acceso solo a personal de la comunidad y evitar duplicidad de información en las bases de datos de la institución.
- **Módulo de gestión de Productos:** Este módulo se compone de toda la información disponible de los productos ya sea desde su creación, eliminación, visualización por parte de los usuarios y actualización pertinente de algún producto, además de la visualización si el producto se encuentra disponible para ser reservado o comprado.
- **Módulo de gestión de Pedidos:** Este módulo permite reservar el pedido, hacer un nuevo pedido, y ver la lista de su pedido. No obstante, este módulo cuenta con los siguientes submódulos:

- **Gestión de Notificaciones:** Este módulo permite a los administradores de la cafetería poder visualizar de forma más rápido y oportuna cuando les realicen un pedido nuevo, para darle una rápida atención.
- **Gestión de pedidos en línea:** Este módulo permite consultar todos los productos ofrecidos en la cafetería, haciendo el pedido desde tu celular.
- **Módulo de Operaciones:** Permite calificar el servicio sobre los pedidos que se han realizado, buscando la forma de mejorar permanentemente, ver las promociones que sean ofrecidas por parte de la cafetería en sus horas felices.
- **Módulo de Gestión de menú:** Este módulo permite cambiar el menú semanal o diario según lo requerido por la cafetería, arma el menú y visualizarlo por parte de los usuarios

1.5 Limitaciones

El desarrollo de este proyecto, se ve limitado en domicilios dentro de la institución puesto que las cafeterías actualmente contar con este servicio les sería muy complicado, además de pagos diferentes a efectivo, por la plataforma que requiere, la comisión que se maneja por cada transacción que se realiza y el costo de mantener este servicio.

Capítulo II

2.1 Marco Referencial

2.1.1 Marco Contextual

La Institución Universitaria Antonio José Camacho (UNIAJC) es una entidad pública es adscrita al Municipio de Santiago de Cali, que ha ido aumentando la cantidad de programas académicos que ofrece, siendo uno de sus fuertes la gran diversidad de horarios. Esto permite que la población del Valle del Cauca vea a la institución como una opción factible para su estudio. Adicionalmente su gran compromiso social al orientar los servicios a las poblaciones con

mayores limitaciones socioeconómicas, hacen aún más atractiva a la UNIAJC. Actualmente la institución cuenta con una población de 7.500 estudiantes aproximadamente.

Dentro de la UNIAJC existen diversos grupos de investigación, como es el caso de GrintTIC. El grupo se compone de diversos semilleros específicamente It Media dentro de este semillero se encuentra el proyecto Smart Campus que busca desde diferentes ejes mejorar la vida de los estudiantes de la Institución con unas subcategorías específicas o ejes donde desarrollar los proyectos relacionados con el mejoramiento institucional, estos son:

- **Smart Living:** Desarrollar servicios que fomenten la integración social, la inclusión, el bienestar, la cultura y la empleabilidad
- **Smart Goverment**: Busca la participación de toda la comunidad y la transparencia de la política y la toma de decisiones.
- **Smart Enviroment**: Centra sus esfuerzos en el uso de las TI para optimizar, gestionar y racionalizar correctamente los recursos naturales.
- **Smart People**: Se enfoca en las habilidades, trabajo y teletrabajo mediante las TI, disponer de acceso a la educación, a la enseñanza y aprendizaje continuo tanto a estudiantes como a personal de la propia universidad, sociedad inclusiva que mejore la creatividad, la innovación y la participación en la vida pública.
- **Smart Economy**: Se enfoca en la educación como recurso base de la economía de la universidad, fomentando la competitividad e impulsar el emprendimiento personal, profesional e innovador

Smart Mobility: Tiene como objetivo gestionar de manera sostenible, segura y eficiente la accesibilidad y los transportes de la universidad.

El ámbito de desarrollo del proyecto es el Smart Living, cuyo objetivo principal es impulsar estilos de vida saludable, comportamientos y consumos. Es así como surge la idea de implementar el proyecto Cafetería Virtual UNIAJC, para facilitar el servicio de reservas de almuerzos, solicitud de refrigerios para reuniones y compra de snacks a la comunidad académica desde la comodidad de su puesto de trabajo o desde cualquier área del campus donde se cuente el estudiante, buscando que la alimentación que es ofrecida a la comunidad estudiantil, directivos, entre otros, incentive el estilo de comida saludable dejando un poco de lado los productos que no tiene beneficio para nuestros sistema. Actualmente, no existe este servicio ofrecido por medio de

aplicaciones a nivel de otras universidades. Mediante una encuesta realizada a nivel nacional, permitió identificar que la mayoría de estudiantes están interesados en tener una aplicación así en su universidad, tenemos otras aplicaciones como uber eat, rappi entre otras, pero estas aplicaciones son enfocadas hacia otro mercado, no suplen las necesidades que enfrentan los universitarios de la institución.

2.1.2 Marco Teórico

2.1.2.1 Proceso desarrollo Smart Campus

Para el desarrollo del proyecto se tiene como modelo el proyecto Smart Campus que provee un desarrollo basado en el trabajo de Pastrana et al (2017) cuyo objetivo principal es tener entregas continuas de alta calidad enmarcadas en un proceso iterativo incremental, con capacidad de mejora continúa basada en revisiones constantes. De esta manera se garantiza la satisfacción de los planteamientos del proyecto. Para llegar a este objetivo las metodologías agiles como es Scrum, propone un modelo con entregas puntuales en un periodo de tiempo denominado sprint que va de 1 a 4 semanas. En él, se realiza el desarrollo de las funcionalidades consolidadas en el artefacto principal del análisis de requisitos llamado product backlog. Este artefacto está constituido por historias de usuario (HU), que utilizan una frase guía para la recopilación de la información de manera fácil. El modelo guía de estas HU es: *yo [un papel]+ de búsqueda [funcionalidad]+ para que [motivo] + Criterios de aceptación* basada en Pastrana et al.(2019)

Una vez consolidadas las necesidades funcionales del proyecto, se sigue rigurosamente el proceso de diseño de software, donde a través de la definición de atributos de calidad y patrones de diseño, se identifican correctamente los drivers que guían la arquitectura para así poder definir correctamente a través de lenguaje unificado modelo (UML), diagramas que reflejen la solución a implementar. Con esto se asegura la escogencia del lenguaje de programación, de los frameworks requeridos y su correcta utilización.

Continuando con el proceso de desarrollo, Smart campus dentro de la etapa de construcción centra su esfuerzo en que la calidad sea ejecutada al mismo tiempo que se desarrolla. Para esto, basado en Pastrana et al (2019), se realiza un control de versiones que permite un seguimiento a los cambios que se le han realizado a el código durante su evolución. La herramienta seleccionada para esto es Gitlab. Sumado a esto, y bajo sugerencia del trabajo de Pastrana et al., (2019), se cuenta con Jenkins para integración continua y Sonar para Análisis Estático de Código (AEC).

2.1.3 Marco Conceptual

- **TypeScript:** Según Angular (2019), se define como un lenguaje de programación que es muy favorable para proyectos muy grandes por lo robusto que es en su estructura de programación basado en

- **XML:** Según Lalmas (2011), es un metalenguaje utilizado para almacenar datos en forma legible y permite manejar grandes cantidades de información.

- **API:** Según API (2007), conjunto de código y funciones específicas que las aplicaciones pueden consumir para poderse comunicar entre ellas.

- **Versionamiento:** Según Pastrana et al (2019), herramienta usada para tener el control del código y registros de cambios cada vez que se realiza modificaciones para el mejoramiento del aplicativo y para saber, así como está el estado de dicho proyecto en desarrollo.

- **Framework:** Según Xie (2016), estructura conceptual normalmente utilizada para software concreto que sirve para el desarrollo del mismo e implementar sus necesidades de uso.

- **Web Service**: Según Juneau & Juneau (2018), intercomunicación entre el servicio y el aplicativo el cual está conectado por la red, por lo general están basado en interacciones

entre solicitudes y respuestas para ello esto vendría siendo (cliente, servidor) el aplicativo cuenta con un servicio SOAP para el login y un REST para el resto de servicios de la aplicación, los cuales usan un protocolo HTTP**.**

- **Notificaciones push**: Según Pedrero-Esteban & Herrera-Damas (2017), envió de alerta al usuario para notificar algún cambio, actualización o pedidos que tenga en el aplicativo**.**

2.1.4 Antecedentes

Desde hace unos años se ha visto un bum sobre los servicios de compra a domicilio con aplicaciones como: Rappi, Uber Eats, Ifood, Domicilios.com, entre otros, que buscan una mayor facilidad y comodidad para los usuarios que utilicen estas plataformas. A continuación, se detallan un poco más:

- **Rappi:** Es una de las principales aplicaciones utilizadas por miles de usuarios en Latinoamérica, fundada por Simón Borrero, donde pueden pedir la comida que deseen a cualquier restaurante o supermercado que se encuentre asociado con ellos, apareció en 2015, nació como una aplicación que ofrecía a domicilio productos de una tienda de barrio. Según cuenta Simón Borrero, tuvieron la suerte de dejar un campo abierto donde la gente podía escribir sugerencias. De pronto empezaron a ver que la gente pedía cosas como hamburguesas, crepes, medicinas y mercado. Pero también se dieron cuenta de que la gente pedía favores. “Lo valioso de Rappi no fue idea de nosotros, fue idea de los usuarios”, confiesa Borrero.
- **Uber Eats:** Es una plataforma de pedidos y entrega de alimentos en línea lanzada por la gigante Uber en 2014 bajo el nombre de Uber Fresh para un año después pasarse a llamar UberEats, moviéndose en las principales plataformas Android e IOs
- **Ifood:** Opera suministros alimentos mediante las plataformas Android e IOs, pudiendo pedir a domicilio sus restaurantes favoritos la comida que deseen, siendo una empresa líder en América Latina con una gran presencia en Argentina, México y Colombia.

Fundada como una puesta en marcha recibió inversiones de capital de riesgo, incluidos los grupos Jorge Paulo Lemann, siendo adquirido en 2014 por Movile, el mismo año en que se produjo la fusión con RestauranteOnline, poniendo el valor de la nueva compañía en aproximadamente unos mil millones de reales. Desde entonces la compañía ha adquirido otro en la industria, y en 2018, la empresa matriz de iFood era parte de la rama de la fintechs

- **Domicilios.com:** Fundada en el 2007. Para ese entonces, José Guillermo Calderón y Miguel McAllister, estudiantes de la Universidad de los Andes, crearon su primer piloto: un directorio digital en el que se listaban todos los restaurantes en las diferentes ciudades de Colombia, La idea fue tomando fuerza cargada con un inusitado impulso buscando innovar en el sector vendiendo por internet. La empresa ha levantado cerca de US$40 millones en capital y hoy en día genera cerca de 500 empleos directos y 1.000 indirectos, siendo lideres con Merqueo.com en la compra de cualquier producto de un restaurante que desees hasta tu casa.

Capítulo III

3.1 Metodología

La metodología utilizada en este proyecto es la propuesta dentro del proyecto de investigación Smart Campus, que utiliza los eventos, roles y artefactos del framework Scrum y las buenas prácticas del framework XP para garantizar la calidad como eje transversal de todo el proceso Pastrana et all (2019).

Framework SCRUM: es un marco de trabajo usado para gestionar el desarrollo de los productos complejos desde los años 90. Scrum no es un proceso o una técnica para construir productos, es un marco de trabajo liviano, fácil de entender, haciendo entregables por sprint que exista el menor reproceso posible Bourque & Fairley (2014).

Se implementarán los siguientes eventos y artefactos por etapa del proceso definidos por La guía definitiva de Scrum: Las reglas del juego (2018), Jeff Sutherland Kent S.

Análisis y Planeación:

Artefactos:

- **Sprint backlog:** listado de funcionalidades del proyecto a realizar dentro de un sprint.
- **Product backlog:** listado de todas las funcionalidades del producto basado en Schwaber & Sutherland (2013).

Eventos:

- **Estimación Poker Scrum:** seria de cartas de la serie de Fibonacci que sirven para darle peso a las historias de usuario
- **Priorización MoSCow:** combinación de letras de: debe tener (M), debería tener (S), podría tener (C), tendrá en esta ocasión (W)
- **Planning meeting:** reunión de planificación del sprint

Diseño:

Artefactos:

- **DAS:** documento arquitectura de software donde se definen las vistas involucradas en el proyecto.

Desarrollo y calidad:

Artefactos:

- **Release:** plan de proyecto conjunto de historias de usuario agrupadas

Buenas Practicas:

- **Daily Scrum:** reunión diaria buscando facilitar la transferencia de información y el apoyo entre los colaboradores del equipo, definiendo 3 puntos básicos, que has

hecho desde la última reunión, que se va a hacer a partir de este momento y que impedimentos se han tenido basada en Schwaber & Sutherland (2013).

- **Integración continua:** los desarrolladores realizan cambios sobre la última versión, cada vez que se realice una modificación se debe estar en la capacidad de integrar con la versión anterior, actualizada y sin errores basada en (Pastrana et al., 2019).
- **Versionamiento:** poder tener acceso al código en cualquier momento, además de tener propiedad colectiva del código al ser accesible por los demás desarrolladores basada en Pastrana et al (2019)
- **Propiedad colectiva del código:** los desarrolladores tienen acceso a todo el código, tienen permitido modificarlo siempre y cuando sea para mejorarlo, corregir errores o agregar funcionalidades basada en Pastrana et al (2019).
- **Ritmo sostenible:** su objetivo es mantener un ritmo de trabajo constante, evitando sobrecargar al equipo. El trabajo extra puede influir de forma negativa en el desarrollo del producto basado en Pastrana et al (2019).

Despliegue:

Artefactos:

- **APK:** entregable para comprobar todas las funcionalidades definidas en la fase de análisis y planeación, se instala en el dispositivo de los interesados.

Eventos:

- **Retrospectiva:** técnica donde se busca concientizar los puntos fuertes del sprint, cosas a mejorar, bases que se tenían.

Capitulo IV

4.1 Desarrollo del proyecto

4.1.1 Análisis

Para el proceso de levantamiento de requerimientos se realizó una reunión en donde se aplicó la técnica Inception Deck. Se utilizaron 8 dinámicas de las 10 sugeridas por (Enrique Comba R, How to star a Project). El objetivo de esta técnica es que todos los asistentes tengan la misma visión de lo que se quiere lograr con este desarrollo. Para su ejecución, se definen unas normas y unos tiempos por cada uno de los tableros que se vayan a utilizar en las dinámicas, es muy importante que se cumplan estos puntos.

Las actividades que se desarrollaron en la reunión fueron:

- **¿Quién está en el auditorio?:**
  Con esta actividad se pretende conocer quienes están participando en el taller. Se trata de identificar: nombre, como contactarlo, que le gusta y que no le gusta. Las personas que participaron fueron:
  - Interesados:
    - Dueños cafetería sede norte
    - Alumno: Fernando Andrés Cifuentes Calderon
    - Alumno: Ronal Andrés Tamayo Zapata
    - Juan Dhalin Rueda
  - Propietaria del producto:
    - Docente: Ana Milena Rojas Calero
  - Scrum Master:
    - Docente: Manuel Alejandro Pastrana Pardo
  - Scrum Team:
    - Alumno: Nicolás Bermúdez Rodríguez
      Alumno: Nicolás Ortiz Acevedo

- **¿Por qué estamos aquí?:**
  Con esta actividad se pretende obtener una visión general del proyecto. Permite que los participantes expongan el motivo por el cual están en la reunión y la información que tienen acerca del proyecto. En donde se define que: En la institución, en el desarrollo de las actividades cotidianas se requiere una herramienta que permita una mayor facilidad en el consumo de los productos ofertados por la cafetería de la institución, integrada con el proyecto de investigación Ecosistema Smart Campus. La disposición de tiempo en ocasiones para tener estos servicios es limitada, dificultando el correcto desarrollo de las actividades cotidianas.

- **Elevator Pitch:**
  Tratar de venderles en un tiempo muy corto la idea del proyecto dándoles datos concisos sobre el proyecto. En él se especifica:
  - ¿Para quién está dirigida la solución?
    Para toda la comunidad perteneciente a la institución
  - ¿Quién está identificando que necesidad u oportunidad?
    - Alumno:
    - Poder realizar comprar de los productos de la cafetería sin necesidad de moverse de su salón.
    - Poder reservar su almuerzo con el fin de no quedarse sin almorzar por los horarios que tenga.
    - Mayor información sobre los productos ofertados por la cafetería.
    - Dueño cafetería:
    - Poder llegar a ofrecer sus productos a toda la comunidad de la institución
    - Incentivar el uso de una nueva modalidad de pedidos institucionales

- Nombre del producto: Cafetería UNIAJC; Frase: Le damos gusto a su gusto; Categoría del producto: Aplicación móvil
- Beneficios que trae la construcción de este proyecto:

- Ofrecer un mejor servicio a la comunidad de la UNIAJC consiguiendo una mayor agilidad a la hora de realizar pedidos de almuerzo, snacks en línea
- Mayor facilidad de compra teniendo los conocimientos de lo ofrecido por la cafetería con sus respectivos precios, pudiendo comprar lo que desee
- Evitar filas al hacer el pedido en línea, me ahorro el tiempo que tomaría en ir y esperar si hay cola para hacer mi pedido, porque ya estaría separado y tan solo sería recogerlo
- Mayor visibilidad de los productos ofrecidos en la cafetería y el precio de cada uno de estos
- Dar a conocer los menús de almuerzos ofrecidos para ver si es de mi gusto
- Mayor venta por la mejoría del servicio al pedir en línea, el tiempo de espera y el conocimiento de los productos

- **Lista del no:**
  Ayuda al equipo a definir el alcance del proyecto, a determinar realmente que es lo que hará y que no hará el producto. Se identifica:
  - No es compatible con dispositivos Windows Phone
  - No es compatible con PC
  - No permite pagar con tarjeta de crédito
  - No permite cancelar pedidos
  - No permite pedir almuerzos después de las 2:30 p.m.
  - No funciona en dispositivos Android inferiores a 4.1 y IOS 7.1
  - No es compatible con Web
  - No se aceptan compras si se desconoce el valor del billete
  - No hace reportes diarios de dinero

- **¿Qué te impide dormir en las noches?:**
  Son todos los riesgos que se identifican que pueden afectar el desarrollo correcto del proyecto. Tenemos los siguientes:

Se requiere pruebas de concepto sobre los framework y el lenguaje con el que se realizara el proyecto

Tabla 1 Riesgos y estrategia identificadas

Riesgos y estrategias identificadas del proyecto

Riesgos	**Estrategias**
Investigaciones conceptuales	Para el desarrollo de este proyecto se tuvo que aprender IONIC en su versión 3.9, además se realizó la investigación para el proceso de creación y manejo de web service, también para el manejo de las notificaciones push, y sobre todo el manejo de la base de datos la cual es PosgrestSQL
Diseñar la arquitectura	La implementación de los atributos de calidad, patrones de diseño y los diferentes diagramas que explican de una forma muy clara la arquitectura
Pruebas para el aseguramiento de la calidad	Se implementará las pruebas con Katalon, Sonar para mirar la deuda técnica y los bugs de la aplicación, integración continua y despliegue continuo para asegurar que cada cambio que se realice pase por el ciclo de calidad

Como se observa en la tabla anterior podemos observar lo riegos y las estrategias que se tendrán en cuenta para el desarrollo de dicho proyecto y no se encuentre contratiempos

Anexo: (Incepcion deck V1.0)

Con la técnica Inception Deck, se facilita ver el producto desde distintas perspectivas, logrando tener un concepto más aterrizado de lo que se quiere obtener con el proyecto.

El siguiente paso a desarrollar son las historias de usuarios. Primero se inicia por la definición de épicas, que son historias sin criterios de aceptación, esto permite definir rápidamente el mínimo producto viable. A continuación, se presenta el formato bajo el cual se registran las historias de usuario

Tabla 1 Formato Historia de Usuario Epica, Basado en Cohn (2004)

Formato de historias de usuario épica

ID de la Historia de Usuario	HU-Cafetería virtual- 001
Rol	Yo como dueño
Funcionalidad	Requiero actualizar la lista de los productos
Razón	Con la Finalidad de poder renovar el listado del menú

Con el formato anterior se obtendrá una mejor perspectiva de lo que desea el cliente a la hora de funcionalidades

Una vez tenemos las historias épicas, las recopilamos todas en un artefacto denominado Product Backlog de acuerdo a lo sugerido por Sutherland y Kent S. Este artefacto es evolutivo y, por tanto, debe ser completadas las historias adicionándoles los criterios de El conjunto de las historias de usuario conforma lo que se conoce como el Product Backlog.

Tabla 3 Formato Historia de Usuario, Basado en Cohn (2004)

Formato de historias de usuario

ID de la Historia de Usuario	HU-Cafetería virtual- 001
Rol	Yo como dueño
Funcionalidad	Requiero actualizar la lista de los productos
Razón	Con la Finalidad de poder renovar el listado del menú
Criterios de Aceptación:	
1. Habiendo pasado el login de inicio de sesión para administrador y estando en la pantalla actualizar la lista de los productos	
2. Opciones son: Agregar cantidad de mercancía, renovar listado de menú del día, domicilios, sugerencias, cerrar sesión	

3. Cuando seleccione esa opción me muestra el menú establecido y me dice si quiero modificarlo y en un campo debajo inmediato dice guardar la información la cual se conecta a la base de datos y debe tener conexión a internet para poder realizar este cambio

4. Cuando cambio la información del menú me sale un mensaje guardado y cambiado correctamente en caso de que sea exitoso el cambio, de lo contrario saldrá un mensaje indicando ha habido un error " por favor revise su conexión a internet"

5. Se visualiza el menú para el día asociado

Con el formato anterior se obtendrá una mejor perspectiva de lo que desea el cliente a la hora de funcionalidades

4.2 Planeación

4.2.1 Sprint Planning

Ya teniendo los requerimientos se realiza una reunión de planeación del sprint, en la que participan el equipo de desarrollo, scrum master y el product owner, con ello se busca realizar la priorización y la estimación de las historias de usuario, esto lleva a acabo que el equipo de desarrollo conozca que funcionalidades se realizaran primero y si alguno en dicha reunión tiene alguna duda es aclarada para no tener percances más adelante en el sprint

En fase de priorización de las historias de usuario se lleva en conjunto con el product owner el cual toma la decisión y mira la necesidad y la importancia de las funcionalidades que se realizan para el aplicativo, se utiliza la técnica Moscow la cual permite clasificar requisitos en el proyecto:

- ✓ **M**ust Have (Debe tener)
- ✓ **S**hould Have (Debería tener)
- ✓ **C**ould Have (Podría Tener)
- ✓ **W**on't Have (No tendrá)

El product owner asigna una letra de acuerdo a la importancia de la historia que se presenta en ese momento, es muy importante que al momento de realizar la priorización se debe tener en cuenta tres factores los cuales son:

- **Valor**: El dueño del producto debe asegurar la entrega de los productos que ofrezcan mayor valor.
- **Riesgo**: Los productos de mayor riesgo en la lista priorizada del producto se les debe proporcionar una mayor prioridad.
- **Dependencias**: Existen historias de usuario que para su elaboración requieren del desarrollo previo de otra historia. Estas dependencias pueden limitar el desarrollo de las historias de usuario en su totalidad; una de la forma de resolver estas dependencias es dividiendo una historia en varias partes o combinar varias historias de usuario de forma interdependiente.

Realizada la priorización, a continuación, se realiza la estimación del esfuerzo necesario para desarrollar la funcionalidad descrita en cada historia de usuario, para ello el equipo Scrum se asegura la entrega de los requisitos al cliente en forma de historias de usuario aprobadas, estimadas y asignadas

Para la estimación se emplea el Planning Poker, esta técnica permite realizar una asignación de puntos a cada tarea, según el nivel de complejidad considerado por el equipo de desarrollo.

Para la asignación del puntaje, la técnica propone una serie de cartas con los valores 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, “?”, mirando cada desarrollador el puntaje que le da dependiendo que tan complicada sea la HU escogida, sino se llega a un acuerdo se debe realizar el ejercicio hasta que se llegue a una sola decisión de puntos por HU, exponiendo porque cada uno le dio ese puntaje a la HU escogida, para esta actividad se utiliza el Scrum Planning Poker.

De esta reunión se obtiene el product backlog priorizado y estimado, así como también una planeación general del product backlog que reflejada en un cronograma de actividades para la asignación de cada sprint a realizar

4.3 Diseño

En esta etapa se definen las restricciones, los atributos de calidad, las vistas arquitecturales y los patrones de diseño, que se implementarán por parte del equipo de desarrollo, con base a lo establecido por el Product Backlog.

4.3.1.1 Atributos de Calidad

Atributos de calidad son características no funcionales deseables en un sistema de software. Pero no todos los sistemas de software deben tener en cuenta todos estos atributos o cualidades, algunas serán más importantes que otras dependiendo del sistema.

La definición de atributos de calidad que conformaran con el cumplimento de los requerimientos identificados anterior mente, a continuación, se mostraran dos tablas las cuales cada una de ellas tendrá los atributos de calidad observables y la otra los atributos de calidad no observables mostrarán una breve descripción, las tácticas o patrones que permitirán el cumplimento del atributo y la forma de aplicación

4.3.1.2 Patrones de diseño

Los patrones de diseño son soluciones para problemas típicos y recurrentes que nos podemos encontrar a la hora de desarrollar una aplicación.

Aunque la aplicación sea única, tendrá partes en común con otras aplicaciones: acceso a datos, creación de objetos. Podemos solucionar estos problemas utilizando algún patrón, ya que son soluciones probadas y documentadas por multitud de programadores

Tabla 3 Atributos de calidad

Atributos de calidad observables

Atributos de Calidad "Observables"			
Atributo de Calidad	**Descripción**	**Tácticas /Patrón de Arquitectura**	**Donde se aplica**
Funcionalidad	Habilidad del sistema para realizar el trabajo para el cual fue concebido Calero, Moraga, & Piattini, (2010)	Patrón: MVC esta se puede apreciar en el Web Service Mediante este patrón permite que las modificaciones a los servicios web de integración sean más sencillas, dado que van a puntos específicos del código. Estandarización en el nombramiento de tablas, módulos, clases, métodos de clase y variables a nivel de diseño y programación	Propende a disminuir la alta cohesión de componentes para garantizar que todo el sistema es más dinámico y adaptable a las necesidades de expansión según lo requiera el sistema.
Performance	Grado en el cual un sistema o componente cumple con su funcionalidad, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. (IEEE 610.12).	Patrón: Inyección de dependencias se puede apreciar en la App y en el Web Service Mediante este patrón permite suministran objetos a una clase en lugar de ser la propia clase la que cree dichos objetos Nuestras clases no crean los objetos que necesitan, sino que se los suministra otra clase 'contenedora' que inyectará la implementación deseada a nuestro contrato	Se implementa en la capa de servicios suministrando objetos a una clase pudiendo tener varias instancias para futuros cambios
Escalabilidad	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental	Patrón multicapa Se implementa un modelo arquitectural multicapa donde la capa de lógica tiene	Se aplica en la capa de lógica de los componentes del sistema.

	Calero, Moraga, & Piattini (2010) capacidad de mejorar recursos para ofrecer una mejora (idealmente) lineal en la capacidad de servicio	la habilidad de propender a una fácil modificación para pasar de una arquitectura web a una REST por medio de la exposición de servicios web.	

En la tabla anterior se observaron los atributos de calidad observables las cuales son cualidades del software

Tabla 4 Atributos de calidad no observables

Atributos de calidad no observables

Atributos de Calidad "No observables"			
Atributo	**Descripción**	**Tácticas /Patrón de Arquitectura**	**Donde se aplica**
Modificabilidad	Es la habilidad de realizar cambios futuros al sistema. Calero, Moraga, & Piattini (2010)	Patrón: MVC Mediante este patrón permite que las modificaciones a los servicios web de integración sean más sencillas, dado que van a puntos específicos del código. Estandarización en el nombramiento de tablas, módulos,clases, métodos de clase y variables a nivel de diseño yprogramación.	Propende a disminuir la alta cohesión de componentes para garantizar que todo el sistema es más dinámico y adaptable a las necesidades de expansión segúnlo requiera el sistema.
Mantenibilidad	Es la capacidad de someter a un sistema a reparaciones y evolución Largo, Erledy, & Mazo, (2005)	Patrón: MVC Mediante este patrón permite que las modificaciones a los servicios web de integración sean más sencillas, dado que van a puntos específicos del código.	Propende a disminuir la alta cohesión de componentes para garantizar que todo el sistema es más dinámico y adaptable a las necesidades de expansión segúnlo requiera el sistema.

		Estandarización en el nombramiento de tablas, módulos,clases, métodos de clase y variables a nivel de diseño y programación.	
Portabilidad	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de cómputo. Estos ambientes pueden ser hardware Largo, Erledy, & Mazo, (2005)	Se implementa el framework ionic que permite programando en un solo lenguaje usarlo para Android y IOS	Se aplica en disminuir la alta complejidad de los diferentes que se utilizan, integrándose en un solo lenguaje base para móviles

En la tabla anterior se pueden observar los atributos de calidad no observables

Se define la arquitectura, para el desarrollo del proyecto es una arquitectura multi capas, se divide en 3 capas:

- **Modelo:** Provee la base necesaria para la manipulación de los datos de la aplicación, además parte del modelo se lo puede usar como clases POCO (Plain Old CLR Objects)
- **Vista:** Parte visual de nuestra aplicación, no teniéndose que ocupar en ningún momento en el manejo de datos, tiene un rol activo
- **VistaModelo:** Capa intermedia entre el modelo y la vista, procesando todas las peticiones que tenga la vista hacia el modelo, además de tener que ocuparse de manejar las reglas del negocio, la comunicación con aplicaciones externas o consumir datos desde alguna fuente

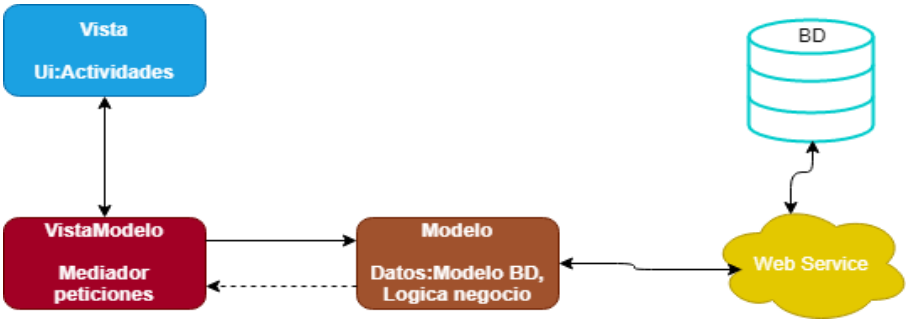


Ilustración 1 Arquitectura MVVM Aplicación Comunicación Académica. Fuente propia

4.3.2. Arquitectura

En la definición de la arquitectura, se seleccionan las vistas que permitirán demostrar el cumplimiento de los atributos de calidad identificados anteriormente.

4.3.2.1 Vista Física

Diagrama de despliegue

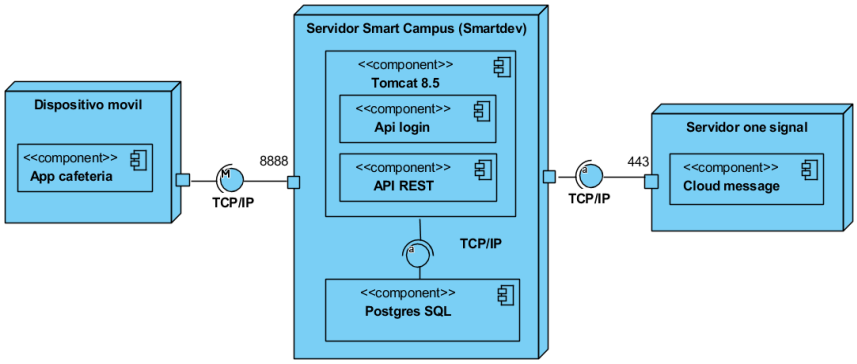


Ilustración 1 Vista física (diagrama de despliegue). Fuente propia

En esta vista la solución está compuesta por dos partes una que contiene el backend y otra que contiene el frontend.

El backend está compuesto por un API rest que contiene los servicios rest del aplicativo que permite dar soporte a la lógica de negocios. Este api está construido en Java con maven utilizando spring. La base de datos es Postgres. La infraestructura que soporta el backend está compuesta de los servidores http://smartdev.uniajc.edu.co:8888 pertenecientes a la UNIAJC. En el http://smartdev.uniajc.edu.co:8888 donde corre un apache tomcat 8.5.30 y en smartdev.uniajc.edu.co la instancia de base de datos.

Las pantallas que se ejecutan a través del apk en Android o la app de Apple consumirán la lógica expuesta en el back a través de los endpoint que se establecieron, permitiendo hacer uso de todos los métodos que se definieron a nivel de back.

4.3.2.2 Vista de Desarrollo

Diagrama de Componentes

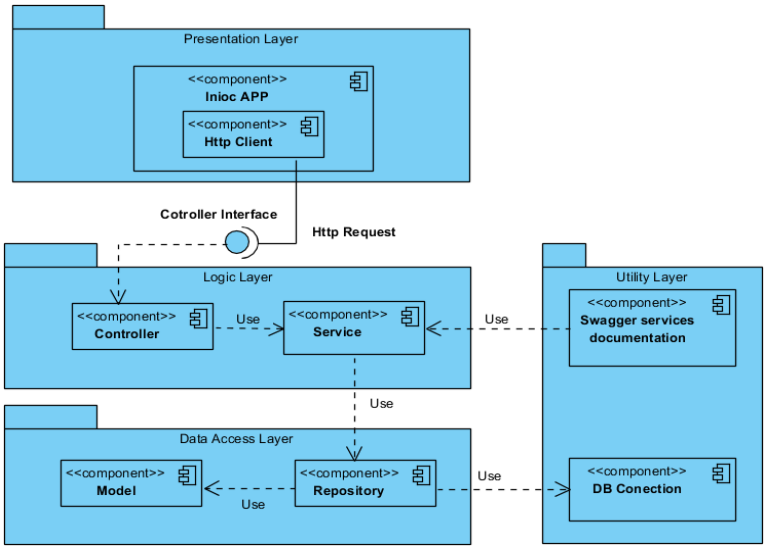


Ilustración 3 Vista de desarrollo (diagrama de componentes). Fuente propia

La arquitectura de la aplicación está dividida en frontend correspondiente a las pantallas que se ejecutan en el dispositivo móvil y en backend que es la lógica de negocios expuesta a través de un API rest de servicios. La estructuración de los servicios se encuentra compuesta por las capas de acceso a datos y capa lógica, además de exponer una interface para recibir la petición. En la capa de acceso a datos se establece la conexión con la base de datos. La comunicación entre capas se da a través de inyección de dependencias. Con respecto al front la comunicación se realiza mediante peticiones HTTP que consumen los endpoint expuestos los servicios del back. Para el módulo de logueo se utiliza el api soap de la UNIAJC garantizando la reutilización de los usuarios ya existentes en la plataforma

4.4 Construcción

Para iniciar la fase de construcción de la aplicación móvil se tiene en cuenta los documentos base el product backlog y el documento de arquitectura de software. Para empezar, se realiza un sprint 0 durante este sprint se configura todo el ambiente desarrollo, donde se requiere instalar:

- Ionic v 3.9.2
- Visual Studio Code v 1.36.1
- Manejador de paquetes NPM v 6.9.0
- Postgresql v 4.0
- Node JS v 8.11.4
- Git
- Spring Tools v 4.1.0.
- Cordova CLI: 8.1.2
- Android SDK Tools: 26.1.1

Una vez configurado el ambiente de desarrollo, se procede a la construcción del software. Se tomó lo decisión de desarrollar el web services utilizando Spring Boot con el framework Spring Tools Suite (STS) para la utilización se hizo con Spring Initializr para tener un diseño inicial de la estructura del proyecto.

Por otra parte, con la creación del web services se puede tener todas las peticiones para tener la arquitectura descentralizada, las peticiones tienen un promedio de respuesta 0.05- 0.03 segundos dependiendo del tipo de conexión que se utilice cuando ya esté integrado con el front.

Debido a que estas herramientas son gratuitas se pueden utilizar sin ningún tipo de problema de pagos, para la mejora de las notificaciones se utilizó un servicio con Node para él envió de notificaciones push, utilizando un demonio que siempre este escuchando las peticiones que se realicen al servicio.

Se adjunta el diseño de las carpetas del servicio web el empaquetado:

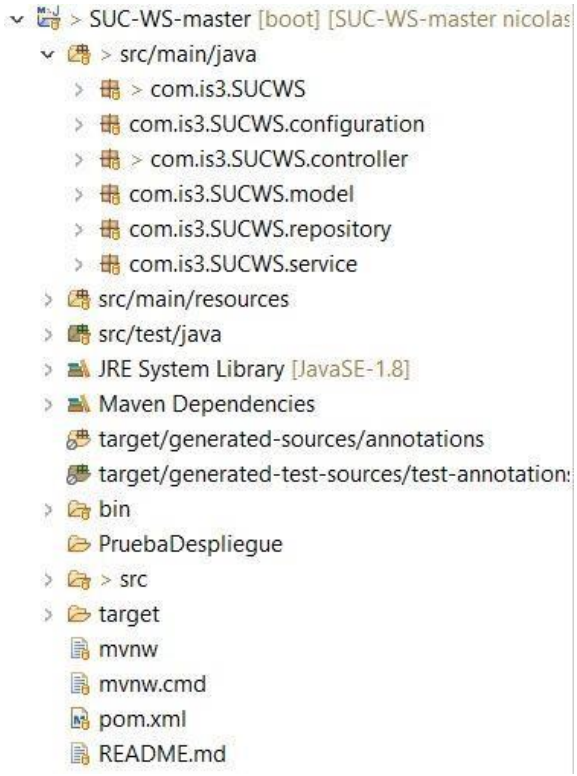


Ilustración 4 Diseño web service Fuente propia

Contextualizando la imagen vista con anterioridad tenemos el modelo donde se crea el llamado a las tablas pertenecientes a la BD y se crean los métodos set and get para poder ser utilizados en otro paquete, también tenemos el repository que es donde se hace

el llamado a la clase del modelo mencionada anteriormente y se crean los métodos para hacer búsquedas de los métodos o query para ser utilizados por el servicio, el servicio es donde se crean los métodos que van a ser utilizados por el controlador para ser expuestos como endpoint, pero la diferencia con el servicio es que solamente se crean pero no se exponen sino que se sigue manejando dentro de la misma RestApi, el controlador es donde se exponen todos los métodos creados en el servicio y se crean los endpoint de consumo para el front, o hacer peticiones en caso que se requiera.

Para tomar la decisión de realizar los servicios web con estilo de interacción Rest y no basados en protocolo SOAP se realizó la siguiente comparación expuesta en la Tabla 5.

Tabla 5 Comparación entre servicios SOAP y servicios Rest. Fuente propia

Comparación entre servicios

	SOAP	**REST**
Basado en	Protocolo de acceso a objetos Simple	Estilo arquitectural de software, Transferencia de Estado Representacional
Exposición de funcionalidades	WSDL, Contrato del Servicio	URI, Identificador Uniforme de Recurso
Estructura del Mensaje	Definido, Basado en XML	No definido, ejemplo de una posible estructura: objeto JSON
Requiere análisis sintáctico del mensaje	Si	No
Protocolo de Transporte	HTTP, HTTPS, SMTP	HTTP, HTTPS
Operación estándar definida	No	GET, PUT, DELETE, POST, HEAD

La tabla anterior se observa la comparación entre un servicio SOAP y un servicio rest

A partir de la comparación anterior se identifica que los servicios web SOAP se caracterizan por la estructura estándar XML en sus mensajes, esto hace que requieran durante el proceso de comunicación, un análisis sintáctico del mensaje, generando en ocasiones gran

consumo en memoria, computación, ancho de banda y recursos. Mientras los servicios REST, al no tener una estructura de mensaje definida, es posible utilizar objetos JSON, y de esta forma lograr un procesamiento e intercambio más eficiente.

Los Servicios web REST son una excelente alternativa para aplicaciones móviles debido a su baja latencia, carga de red y el tamaño del mensaje.

Con la investigación realizada por parte de los servicios REST y SOAP se tomó la decisión de utilizar los servicios REST porque son más óptimos en el estilo del proyecto que se utilizara.

Tabla 6 Descripción de las carpetas/archivos API REST. Fuente propia

Descripción de las carpetas

Carpeta/Archivo	**Descripción**
Controller	Contiene lo heredado por el servicio y es donde se exponene las peticiones creadas en el service para ser consumidas desde el front
Model	Contiene los modelos que van a ser utilizados por el repository utilizando la herencia para la creación de la búsqueda por id, nombre, además de utilizar sus get y set respectivos
Repository	Se utiliza la herencia de clases para coger todos los datos instanciados por el modelo, haciendo un filtrado por id, nombre, date, etc.
Service	Se encuentran la herencia del repository que a su vez hace una herencia de model pudiendo utilizar todos los datos instanciados y creando las peticiones para ser heredadas por el controller.
Pon.xml	Este archivo es donde se contiene toda la configuración para generar el archivo .war y los demás como las dependencias de tomcat, postreSQL, JPARepository, Maven, entre otros.

En la tabla anterior se observa la descripción del contenido de cada una de las carpetas o archivos que con tiene un API REST

4.4.1 Desarrollo de Aplicación Móvil

4.4.1.1 Android y iOS

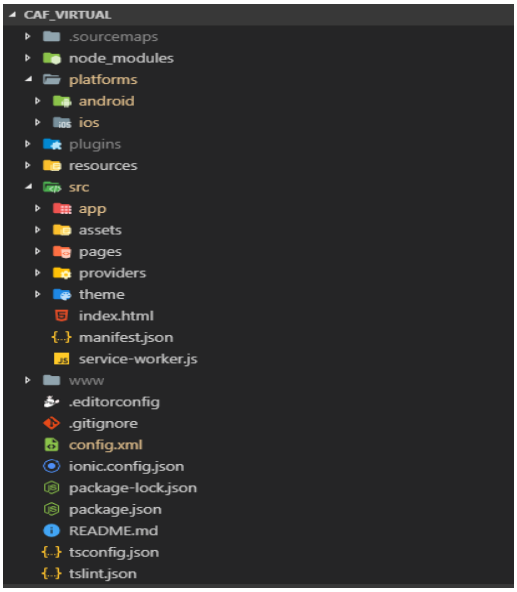


Ilustración 5 Diseño carpetas Ionic. Fuente propia

Tabla 7 Descripción de las carpetas/archivos Ionic. Fuente propia

Descripción de las carpetas en IONIC

Estructura del Proyecto Android y IOS	
Carpeta	Descripción
sourcemaps	Es un mapeo que se realiza entre el código original y el código transformado, tanto para archivos JavaScript como para archivos CSS
Node modules	Contiene todas las librerías que se están usando en el proyecto y también librerías propias de ionic
platfmors	Contiene dos carpetas las cuales una es Android y la otra IOS
Android	En esta carpeta se encuentra todo lo necesario para que Ionic realice su compilación completa y pueda generar una apk en Android

IOS	En esta carpeta se encuentra todo lo necesario para que Ionic realice su compilación completa y pueda generar una apk en IOS
plugins	Se encuentra los plugins utilizados, en este caso se tomara uno como ejemplo el uso de la cámara (cordova-plugin-camera)
resources	La cual tiene dos carpetas cada una de ellas dice Android y ios estas contiene el Icon y el splash correspondiente a cada plataforma
src	Contiene todo las paginas necesarias para el funcionamiento de la aplicación, modelos, lógica del negocio
app	Se configura todo los accesos cuando la aplicación se lanza, y lo que se visualizara por parte la app visual
assets	Contiene las imágenes más importantes del aplicativo
pages	Contiene todas las páginas de navegación del aplicativo
providers	El cual contiene un servicio que se vaya a realizar con la conexión a la base de datos
theme	Tiene variable globales para el css y también contiene el dark theam y light theam
www	En esta carpeta se encuentra todo lo necesario para que Ionic realice su compilación completa y realice una versión web total mente funcional

En la tabla anterior se observa como describe el contenido de cada una de las carpetas de IONIC

Modelo Vista – Vista Modelo

Para el desarrollo del proyecto se tiene en cuenta la arquitectura definida mediante el patrón modelo vista – vista modelo. La arquitectura MVVM tiene por objeto simplificar las tareas de desarrollo y mantenimiento de software escrito a través de las divisiones de ocupaciones. Adicionalmente, se puede usar con múltiples vistas sin tener que modificarlas. Este modelo permite que se pueda trabajar en paralelo un cambio en la vista no provoca que se tenga que modificar la capa “vista modelo”. Con esto se puede obtener una mejor reusabilidad, testabilidad.

Beneficios de MVVM: Durante el desarrollo, las actividades y diseño se llevan a cabo de forma más independiente y concurrente en sus componentes.

Se pueden crear pruebas unitarias para el modelo y los view models sin utilizar la vista. Esta podría ni existir y aun así se podrían desarrollar el modelo y los view models completamente.

Es fácil rediseñar la interfaz de usuario de la aplicación sin tocar el código porque las vistas están implementadas completamente en XAML.

- **Modelo**: Provee la base necesaria para la manipulación de los datos de la aplicación, además parte del modelo se lo puede usar como clases POCO (Plain Old CLR Objects)
- **Vista**: Parte visual de nuestra aplicación, no teniéndose que ocupar en ningún momento en el manejo de datos, tiene un rol activo
- **VistaModelo**: Capa intermedia entre el modelo y la vista, procesando todas las peticiones que tenga la vista hacia el modelo, además de tener que ocuparse de manejar las reglas del negocio, la comunicación con aplicaciones externas o consumir datos desde alguna fuente (Model-View-ViewModel (MVVM) Design Pattern using Windows Presentation Foundation (WPF) TechnologyErik Sørensen, Marius Iulian Mihailesc).

Con lo anteriormente mencionado se puede determinar que con este modelo es más fácil la mantenibilidad del código, si se fija en la ilustración 5, se puede ver una carpeta la cual es llamada app el cual es que realiza la manipulación de los datos de la aplicación, luego vemos la carpeta pages el cual no proporciona todas las vistas de nuestro proyecto y por ultimo vemos la vista modelo la cual si nos dirigimos a providers este contendrá y realizara las peticiones al servicio del back de nuestro servició además de tener que ocuparse de manejar las reglas del negocio, la comunicación con aplicaciones externas o consumir datos desde alguna fuente

Tabla 8 Descripción de notificaciones push

Notificaciones Push

Dependencias		
Nombre	Descripción	Version
One signal	facilita la gestión de las notificaciones push dentro de la aplicación.	3.11.2

En la tabla anterior se encuentran la descripción de one signal (notificaciones push)

4.4.2 Definición de Tecnologías

En las tablas 9 y 10 se observan las tecnologías utilizadas para el proyecto.

Tabla 9 Descripción de tecnologías App Ionic. Fuente propia

Descripción de tecnologías

Tecnologías App Móvil Android / IOS	
Arquitectura	MVVM
Base de Datos	Postgresql v 4.0
Lenguaje Front End	XML / HTML
Lenguaje Back End	Typescript / CSS
IDE	IONIC
Servidor Repositorio	Gitlab v 10.6
automatización de tests	katalon studio v5.91
Integración Continua	Jenkins v 2.138
Analizador de código fuente	Sonar v 7.1

Tabla 10 Descripción de tecnologías API. Fuente propia

Descripción de tecnologías de API

Tecnologías API	
Arquitectura	MVC
Base de Datos	Postgresql
Lenguaje	Java
Servidor Repositorio	Gitlab
Editor de código	STS

4.5 Pruebas

Con la importancia de probar el API Rest desarrollado se utiliza una herramienta la cual es llamada SOAP UI la cual nos permite probar el funcionamiento del servicio, las características del computar en el cual se realizaron las pruebas son: procesador Inter(R) Core (TM) i5-321M CPU 2.50GHz 2.50 GHz disco duro Kingston SSD, Memoria RAM 8Gb sistema operativo: Windows 10 64 bits

SoapUI proporciona 4 estrategias para las pruebas: carga simple, ráfaga, hilo y varianza. Para este proyecto se usó la estrategia simple, que realizar pruebas de rendimiento básico del servidor e identificar la capacidad máxima de atención de usuarios y peticiones de la aplicación desplegada antes de estresar el servidor completamente. Para las pruebas se seleccionaron aquellos que tendrán mayor demanda por parte de los usuario y administradores los cuales son:

- Listar admin
- Listar sugerencias
- Listas productos
- Listar promociones
- Listar almuerzos

Para la configuración del escenario de prueba inicial se define 10 hilos a usar, una estrategia de tipo simple, un tiempo de mil milisegundos entre cada conjunto de solicitudes a realizar, y un factor aleatorio de carga de 0.5 lo que permitirá distribuir uniformemente las solicitudes entre 500 ms y 1000 ms

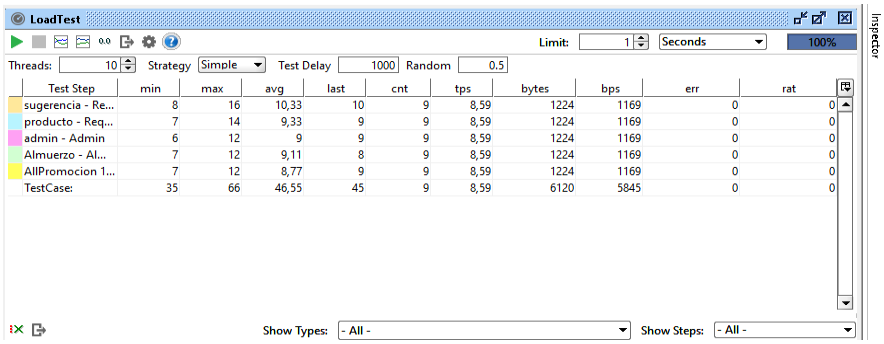


Ilustración 6 Tablero de pruebas de carga SoapUI, resultados de las pruebas del API en SoapUI, Fuente propia

Con el objetivo de aclarar los resultados vistos en la ilustración 6, se definen cada una de las columnas a continuación:

Tabla 11 Descripción de las métricas SoapUI, Fuente Documentación oficial SoapUI

Métricas de SoapUI

Columna	Descripción
Min	Tiempo mínimo medido en milisegundos que tomó la ejecución de la petición.
Max	Tiempo máximo medido en milisegundos que tomó la ejecución de la petición.
Avg	Tiempo promedio medido en milisegundos que tomó la ejecución de la petición.
Last	Valor del tiempo en milisegundo de la última petición realizada.
Cnt	La cantidad de veces que se ejecutó la petición.
Bytes	El número de bytes procesados por el paso de prueba.
Bps	Los bytes por segundos procesados por el paso de prueba.
Err	El número de errores de aserción para el paso de prueba.
rat	Porcentaje de solicitudes que fallaron.

En la tabla anterior se observa las métricas sobre soapui

De lo anterior, es posible concluir el tiempo promedio para la ejecución de cada una de las peticiones realizadas medidas en milisegundos es:

- Listar admin: 6 ms

- Listar sugerencias: 8 ms
- Listas productos: 7 ms
- Listar promociones: 7 ms
- Listar almuerzos: 7 ms

4.5.1 Katalon

¿Qué es katalon?

- Katalon Studio es una herramienta gratuita para automatización de tests sobre aplicaciones web y móviles.

¿Con que está construido este aplicativo?

- está construida sobre Selenium y Appium y ofrece un entorno construido sobre Eclipse para automatizar las pruebas.

¿Para qué sirve katalon?

- La herramienta permite la grabación de una navegación sobre los principales navegadores (Chrome, Firefox, IE, Edge)

Características de Katalon Studio

1- Katalon nos permite automatizar aplicaciones web, aplicaciones móviles

2- Código abierto - Descargue y comience a usar desde este momento.

3- En Build Library que ahorra mucho tiempo para un tester.

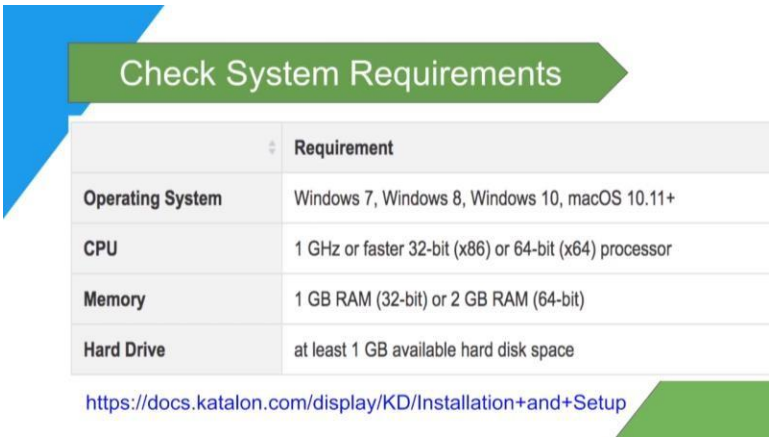
4- Grabación y reproducción para aplicaciones web y móviles.

5- Integración con Gitlab y GitHub, integración de Jenkins, etc.

6-Katalon crea informes automáticamente y podemos exportar el informe en diferentes formatos, como CSV, HTML, PDF, etc.

7-Nos permite enviar un informe por correo electrónico con o sin archivo adjunto. También tiene una plantilla la cual se puede modificar

Requerimientos del computador



Check System Requirements

	Requirement
Operating System	Windows 7, Windows 8, Windows 10, macOS 10.11+
CPU	1 GHz or faster 32-bit (x86) or 64-bit (x64) processor
Memory	1 GB RAM (32-bit) or 2 GB RAM (64-bit)
Hard Drive	at least 1 GB available hard disk space

https://docs.katalon.com/display/KD/Installation+and+Setup

https://www.katalon.com/

Prueba katalon automatización en aplicación de cafetería

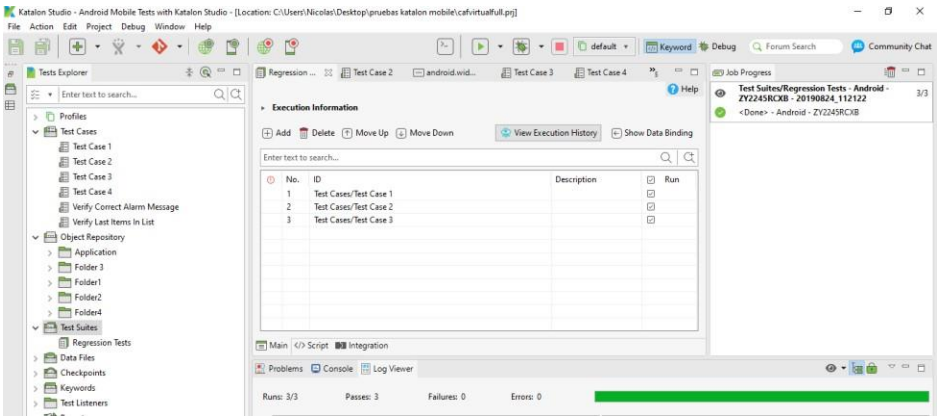


Ilustración 7 Prueba automatización Katalon, Fuente propia

4.5.2 Aseguramiento de la calidad

El desarrollo del proyecto se llevó a cabo, bajo un ambiente de calidad preventiva, en donde se pretende de forma constante mejorar y monitorear el desarrollo del software, realizar una corrección temprana de errores y garantizar el desarrollo bajo estándares internacionales.

Esto se logró gracias al entorno automatizado proporcionado por el proyecto Ecosistema Smart Campus el cual está compuesto por la integración de dos herramientas:

- **Gitlab**: Servicio web de control de versiones de código abierto basado en Git.
- **SonarQube**: Es una plataforma de código abierto, que permite la gestión de calidad de código, analiza y mide la deuda técnica, detecta vulnerabilidades, errores de punteros nulos, comportamiento no deseado.

Ilustración 8 vista general de ambiente de calidad Fuente propia

El proceso del aseguramiento de la calidad inicia una vez se realiza el versionamiento del código en Gitlab.

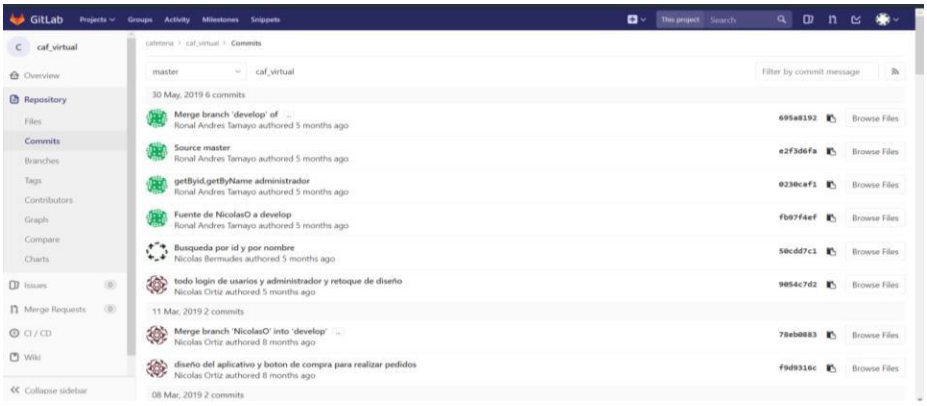


Ilustración 9 Configuración de herramientas aseguramiento de calidad, Fuente propia

A continuación, se presenta el tablero de SonarQube, en el cual es posible observar la calidad del producto, así como la cantidad de bugs y vulnerabilidades, la deuda técnica, proporcionando el porcentaje de duplicación de código. Dentro de las políticas del aseguramiento de calidad las métricas deben disminuir durante el proceso de desarrollo del proyecto.

En la primera evaluación se identifican 0 vulnerabilidades, 2 horas de deuda técnica ocasionada por la detección de malas prácticas, una duplicación de código de 3.3% equivalentes a 6 bloques de código duplicados.

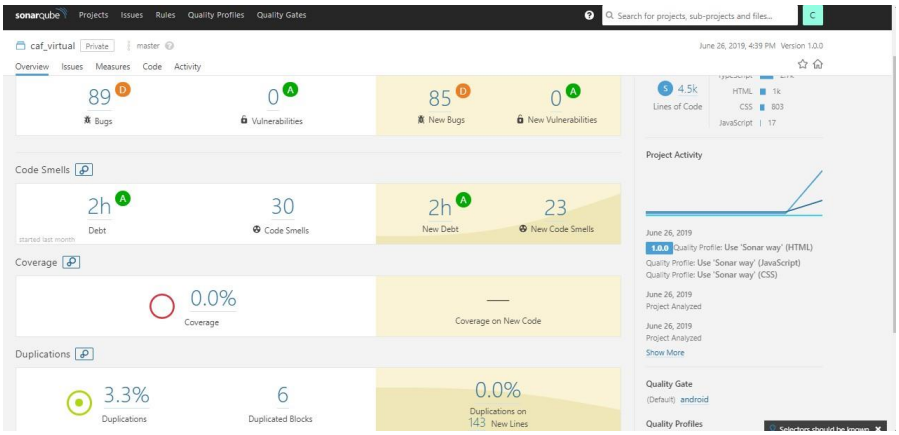


Ilustración 10 Dashboard Sonar Métricas Calidad (Estado Inicial), Fuente propia

En el siguiente análisis se corrige la cantidad de bugs, así como se reduce la deuda técnica a 51 minutos

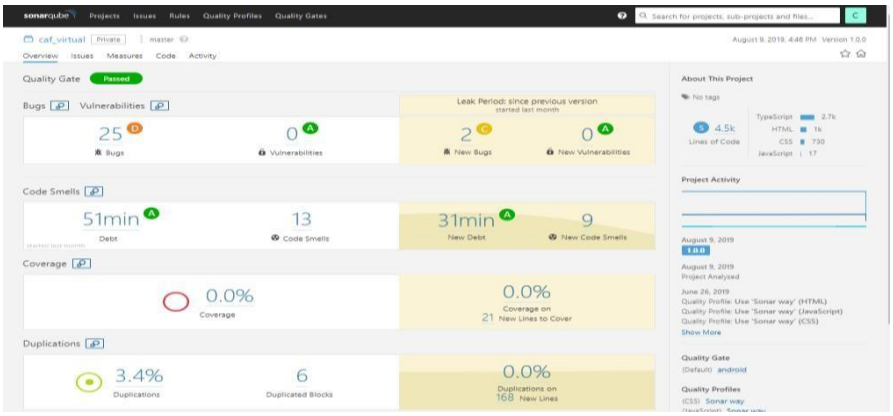


Ilustración 11 Dashboard Sonar Métricas Calidad (Revisión 2), Fuente propia

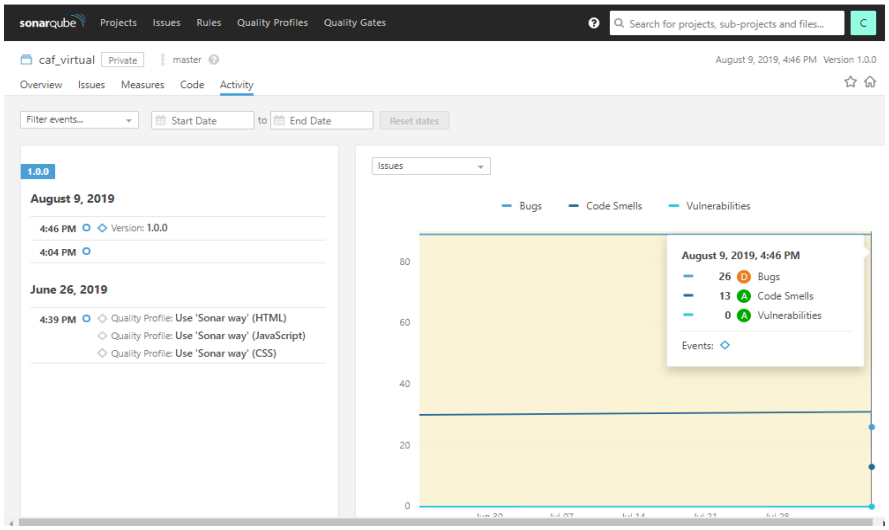


Ilustración 12 Dashboard Sonar Métricas Calidad (Revisión 2), Fuente propia

En el tercer análisis se evidencio una reducción de bugs y deuda técnica casi llegando a menos de 17 minutos

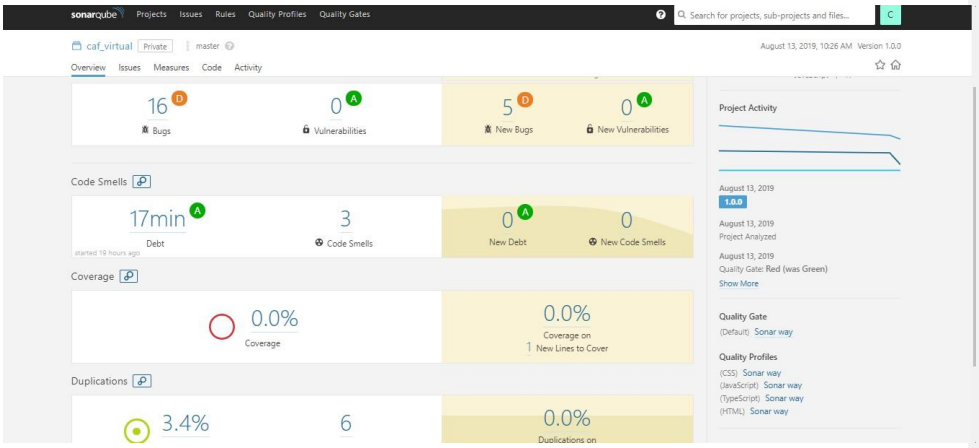


Ilustración 13 Dashboard Sonar Métricas Calidad (Revisión 3), Fuente propia

Al finalizar el proyecto las métricas de calidad son:

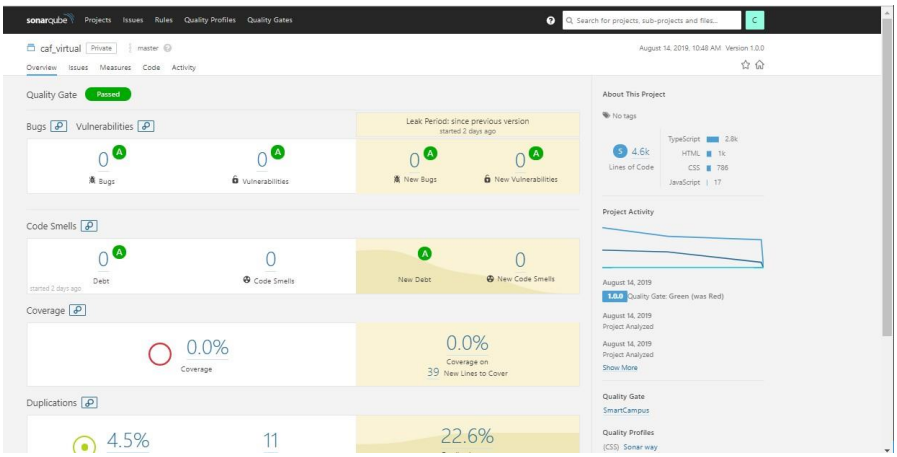


Ilustración 14 Dashboard Sonar Métricas Calidad (Revisión final), Fuente propia

Históricos del análisis estático de código:

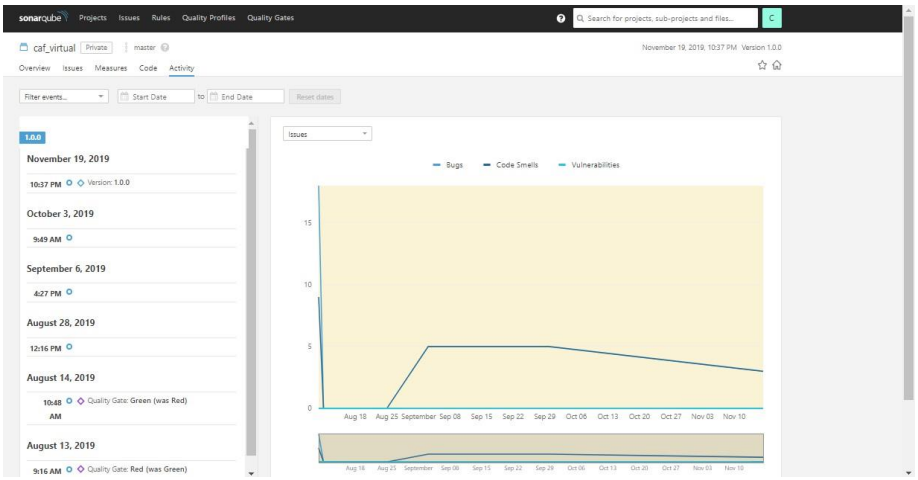


Ilustración 15 Dashboard Sonar Métricas Calidad (Revisión final), Fuente propia

5. Capítulo V

5.2. Conclusiones

Mediante este proyecto es posible evidenciar que el uso de la metodología propuesta por el proyecto Ecosistema Smart Campus, permite tener un orden del ciclo de vida del desarrollo del software, donde en cada etapa se hacen entregas a los interesados del proyectó. Además, permite analizar y diseñar cada requerimiento, buenas prácticas de desarrollo.

La técnica Inception Deck, resultó de gran utilidad para tener todos los interesados del proyecto la misma visión de lo que se quería lograr con el desarrollo del proyecto teniendo desde el principio las limitantes del proyecto, sus restricciones y alcance definidos.

Se logró diseñar arquitecturalmente una solución capaz de cumplir con los requerimientos funcionales. La implementación del patrón arquitectónico MVVM en el desarrollo móvil apoyó a la construcción de un software robusto cumpliendo con los estándares de calidad, teniendo la arquitectura dividida por vistas de las capas de responsabilidad, buscando un óptimo rendimiento y fácil entendimiento del código.

La construcción de un API Rest permite exponer como servicio la lógica del negocio. Esto permite utilizar toda la lógica implementada en un front para su consumo y visualización, de esta forma se tiene una respuesta más óptima comparado con una arquitectura no desacoplada.

Logramos comprender la importancia de calidad preventiva en el desarrollo de los proyectos, ya que se tiene la certeza que el nivel de bugs y vulnerabilidades del proyecto van a ir disminuyendo y esto no va a ser un problema el día de mañana, desarrollando buenos hábitos de programación, apoyados en un entorno automatizado que facilita el control de calidad y el seguimiento del mismo, con las herramientas Gitlab como versionador, Sonar como analizador estático de código.

Para concluir, cuando se maneja un proyecto con unas definiciones claras desde el inicio los requerimientos tienen una mayor posibilidad de éxito, teniendo una arquitectura basada en atributos de calidad y patrones de diseño para determinar que el desarrollo se realice bajo ciertos estándares, con el versionamiento de código y analizador estático de código se tiene un aprendizaje mucho más rápido de los errores que se encuentren, permitiendo solucionarlos de una forma mucho más rápida. La calidad se encuentra en todas las etapas que se vieron cubiertas en

este documento, pudiendo obtener un equipo motivado, organizado y enfocado en lo que se quería lograr.

5.3. Recomendaciones y trabajos futuros

La arquitectura definida para el proyecto facilita la implementación de mejoras y/o cambios futuros, dentro de los cuales se proponen:

- Desarrollar un módulo que permita el pago previo a la compra de un artículo mejorando los procesos de compra, teniendo una mayor seguridad en los productos que se están ofreciendo y la disponibilidad de estos
- Se recomienda el acceso desde web por facilidad de los dueños de la cafetería la interfaz puede resultar más sencilla a la hora de hacer actualizaciones y porque casi todas las cafeterías tienen un computador así sea pequeño sin tener que depender del dispositivo móvil.
- Permitir que los usuarios tengan una administración de sus perfiles puesto que así facilita ver el usuario, los accesos que tengo como usuario y posibles ajustes de la aplicación que actualmente no se tienen centralizados

Dentro de otras recomendaciones están:

- Investigar para poder hacer cambios a versiones más actualizadas de Ionic que permite funcionalidades nuevas que con esta versión no están disponibles
- La calidad integrada con despliegue continuo para mejorar la automatización de esta y no tener que realizar este proceso manual

6. Referencias

Angular. (2019). Angular - Architecture overview.

API. (2007). Recommended Practice for Planning , Designing and Constructing Fixed Offshore Platforms — Working Stress Design. *Api Recommended Practice*. https://doi.org/10.1007/s13398-014-0173-7.2

Calero, C., Moraga, M. Á., & Piattini, M. (2010). Calidad Del Producto Y Proceso Software. In *DFVCL*. https://doi.org/ISBN: 8478979611 ISBN-13: 9788478979615

Cohn, M. (2004). User Stories Applied: For Agile Software Development (Addison Wesley Signature Series). In Writing (Vol. 1). https://doi.org/10.1017/CBO9781107415324.004

Juneau, J., & Juneau, J. (2018). RESTful Web Services. In *Java EE 8 Recipes*. https://doi.org/10.1007/978-1-4842-3594-2_15

Kruchten, P. B. (1995). The 4+1 View Model of Architecture. *IEEE Software*. https://doi.org/10.1109/52.469759

Lalmas, M. (2011). XML information retrieval. In *Understanding Information Retrieval Systems: Management, Types, and Standards*. https://doi.org/10.1007/978-3-642-39314-3

Pastrana, M., Ordóñez, H., Ordóñez, A., & Merchan, L. (2017). Requirements elicitation based on inception deck and business processes models in scrum. *Communications in Computer and Information Science*. https://doi.org/10.1007/978-3-319-66562-7_24

Pastrana, M., Ordóñez, H., Rojas, A., & Ordoñez, A. (2019). Ensuring Compliance with Sprint Requirements in SCRUM: Preventive Quality Assurance in SCRUM. *Advances in Intelligent Systems and Computing*. https://doi.org/10.1007/978-981-13-6861-5_3

Pedrero-Esteban, L.-M., & Herrera-Damas, S. (2017). La notificación push como estrategia informativa de la radio en el entorno digital. *El Profesional de La Información*. https://doi.org/10.3145/epi.2017.nov.09

Pressman, R. S. (2012). Software-Engineering 7th ED by Roger S. Pressman. In Software Engineering A Practitioner's Approach.

Schwaber, K., & Sutherland, J. (2017). La Guía Definitiva de Scrum: Las Reglas del Juego.

Xie, Q. (2016). Conceptual Framework. In *Technical and Vocational Education and Training*. https://doi.org/10.1007/978-3-319-30157-0_2