SISTEMA DE INFORMACIÓN MULTIPLATAFORMA PARA ASOCIACIÓN COMERCIALIZADORA DE AGUACATE HASS EN COLOMBIA

John Eduard Bolaños Camacho

Nilson Giovanni Amaya Santana.

Universidad de Cundinamarca

Extensión Chía

RESUMEN.

En los últimos años la producción y exportación del aguacate Hass ha ido en aumento en el país, por lo cual su constante crecimiento demanda soluciones que optimicen los procesos de recepción y comercialización de dicho producto. Por lo anterior, se presentó un sistema de información multiplataforma para la asociación tolimense AGROBILBAO el cual, luego de su debida recolección de información e identificación de las principales necesidades, se desarrolló una aplicación web y móvil las cuales permitieron facilitar los procesos administrativos, incrementar la comunicación y contar con un adecuado manejo de procesos entre los integrantes de la asociación. Esto fue posible por medio del uso de una metodología de desarrollo de software del modelo espiral alcanzando el desarrollo de una aplicación enfocada a los usuarios asociado y usuarios clientes.

PALABRAS CLAVE: Aplicaciones móviles, Backend, Base de datos, Frontend, Sistema de información, Webservices.

ABSTRACT.

In recent years the production and export of Hass avocado has been increasing in the country, so its constant growth demands solutions that optimize the processes of reception and commercialization of said product. Therefore, a multiplatform information system for the AGROBILBAO tolimense association was presented, which, after its proper collection of information and identification of the main needs, developed a web and mobile application which allowed to facilitate administrative processes, increase the communication and have adequate process management among the members of the association. This was possible through the use of a software development methodology of the spiral model, reaching the development of an application focused on the associated users and client users.

KEYWORDS: Backend, Data bases, Fronted, Information system, Mobile Apps, Web services.

INTRODUCCIÓN

La tecnología ha hecho posible que empresas puedan mejorar procesos internos, permitiendo localizar falencias y errores de comunicación que causan pérdidas en cualquier proyecto. Según datos del Instituto Colombiano Agropecuario (ICA) el aguacate hass ha llegado a incrementar sus exportaciones en más del 413 % en menos de tres años. La mayor cantidad de cultivos con esta producción se encuentran a lo largo y ancho de Antioquia, representa el 52,8 % de la fruta que se vende a otros países. Además, un 80 % de esas 3.916 hectáreas de tierra en la región pertenece a pequeños productores (Suarez, 2019).

AGROBILBAO hace parte de ese 20% restante de dueños productores, ubicando en otro departamento del país. Actualmente este tipo de organizaciones tienen un manejo convencional de la información, se usan libros de Excel muy básicos, utilizan registros de recepción y ventas en hojas físicas; se hace difícil el archivo y almacenamiento de la información, no es organizada, no se tienen datos estadísticos que promuevan a generar acciones de respuesta a los diferentes problemas internos, en el acopio y recepción del producto se evidencian falencias como la falta de organización y comunicación efectiva entre los miembros participes activamente en la organización. Lo que produce perdidas de coste para la asociación y sus afiliados en este momento por las problemáticas detectadas en la asociación objeto de estudio y por su falta de organización interna, al no contar con una herramienta administrativa como esta, donde se puedan tener registradas todas las actividades relevantes que permiten hacer una gestión eficiente.

Con lo anteriormente dicho, la asociación necesita estar a la vanguardia para lograr una distribución y exportación impecable, empezando desde el interior. Un sistema de información representa una herramienta tecnológica adecuada para optimizar los procesos internos de la asociación, permitiendo mitigar los problemas generados por las falencias en organización y falta de comunicación efectiva dentro de la asociación. Por medio del cual los usuarios podrán llevar a cabo las principales tareas y labores en un menor tiempo posible, ya que estarán conectados en tiempo real por medio de las aplicaciones web y móvil, donde podrán acceder a los datos y también generar reportes para la toma de decisiones claves en la Asociación. Varias empresas han desarrollado sistemas de información para el área agricultor con el fin de mejorar la comunicación entre campesinos, empresarios y escuelas; Agronet ofrece un foro para expresar ideas en inquietudes, ofreciendo también capacitaciones sobre el manejo de la tecnología a quienes lo necesitan (Ministerio de agricultura, 2018). BeSoftware es otro sistema de información el cual

suple necesidades como la clasificación de productos, reporte de trazabilidad y costes de producción (Joyanes, 2014). Por otro lado, Agromovil plantea un sistema de información por telefonía móvil, donde pueden consultar precios de productos, el clima, bolsa de oferta y demanda, alertas y asesoría para técnicas de producción y manejo de productos vía telefónica (Agromovil, 2014). Para llevar a cabo un sistema de información que mitigue las falencias que presenta actualmente la asociación AGROBILBAO fue necesario escoger herramientas que permitan desarrollar tanto el aplicativo web como el móvil, como lo son el motor de base de datos, framework a usar, entorno de desarrollo, backend, fronted, librerías, lenguaje de programación, web service e interoperabilidad en dicho proyecto.

a) Pregunta de investigación

¿Cómo optimizar los procesos de recepción y comercialización de aguacate HASS en la Asociación AGROBILBAO, por medio del desarrollo de un sistema de información multiplataforma que permita mitigar las falencias en organización y centralización de los datos?

b) Objetivo General

Desarrollar un sistema de información multiplataforma para optimizar los procesos de recepción y comercialización de aguacate HASS en la Asociación AGROBILBAO.

c) Objetivos Específicos

- Recolectar la información fundamental de la actividad económica efectuada en la Asociación AGROBILBAO, para identificar las principales necesidades que esta presenta en su sector organizacional.
- Desarrollar la aplicación web, que facilite los procesos con agilidad y transparencia en pro de incrementar la comunicación y productividad de los campesinos pertenecientes a la asociación.
- Desarrollar la aplicación móvil con sistema operativo Android, que sirva de complemento facilitador en modo de consulta para el adecuado manejo de los procesos, por parte de los integrantes de la asociación.

• Realizar las pruebas que permitan verificar el correcto funcionamiento de los aplicativos de acuerdo a los requerimientos establecidos para el adecuado manejo de la herramienta.

MÉTODOS Y MATERIALES

La metodología para usar en este proyecto es la espiral propuesta por Barry Boehm la cual se caracteriza porque según Fariño, G. (Bohem. 1988) en su trabajo titulado "Modelo Espiral de un proyecto de desarrollo de software" define esta metodología como "un modelo meta del ciclo de vida del software donde el esfuerzo del desarrollo es iterativo, tan pronto culmina un esfuerzo del desarrollo por ahí mismo comienza otro".

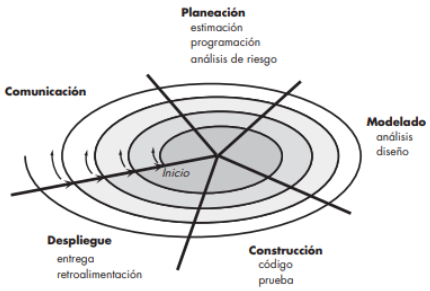


Figura 1. Metodología de desarrollo en espiral.

Para llevar a cabo el desarrollo del sistema de información multiplataforma se llegó a la conclusión que la metodología del modelo espiral era el más indicado para realizar el proyecto por su versatilidad y adaptación al cambio. Esta metodología contó con cinco etapas que se dividen en: comunicación, planeación, diseño y modelado, construcción y despliegue.

1. Fase de Comunicación:

En la fase de comunicación, se realizó el levantamiento de información necesario para el desarrollo del sistema, con la interacción con personal de la asociación y miembros de la misma. Se identificaron las necesidades y falencias que presentan. A continuación, se relacionan los requisitos funcionales y no funcionales en las figuras 2 y 3:

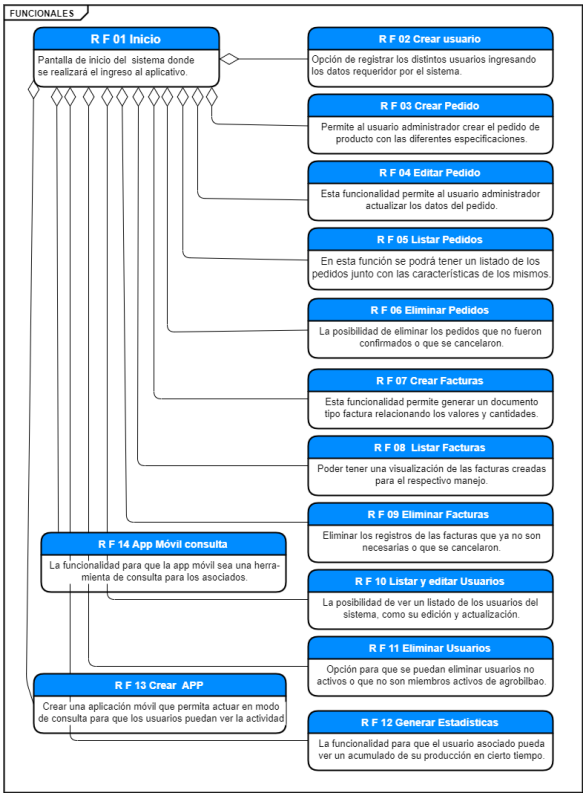


Figura 2. Requisitos Funcionales**.**

Requisitos funcionales: se listan las diferentes funcionalidades que aplican como requisitos funcionales, con las necesidades y especificaciones necesarias para satisfacer las necesidades del cliente.

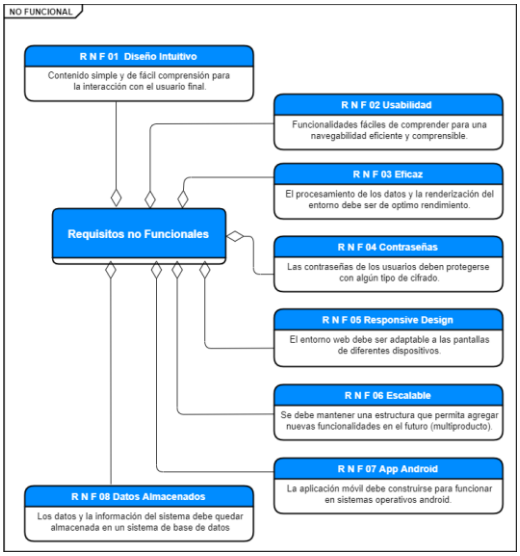


Figura 3. Requisitos no funcionales.

Requisitos no funcionales: se observar aquellos requisitos que no afectan directamente el funcionamiento del sistema, pero que son requeridos por el cliente para una mejor usabilidad por parte del usuario y un mejor rendimiento del manejo de datos.

2. Fase de Planeación:

La construcción del modelo de la base de datos requirió un proceso que describimos en cuatro pasos. Primero contar con unos requerimientos claros, los cuales nos permitieron el segundo paso de tener las entidades y sus respectivas relaciones claras, tercero realizar una tabulación del diseño lógico que permita la normalización de la base de datos, en cuarto y último lugar definimos los tipos de datos y la decisión de utilizar PostgreSQL como motor de base de datos.

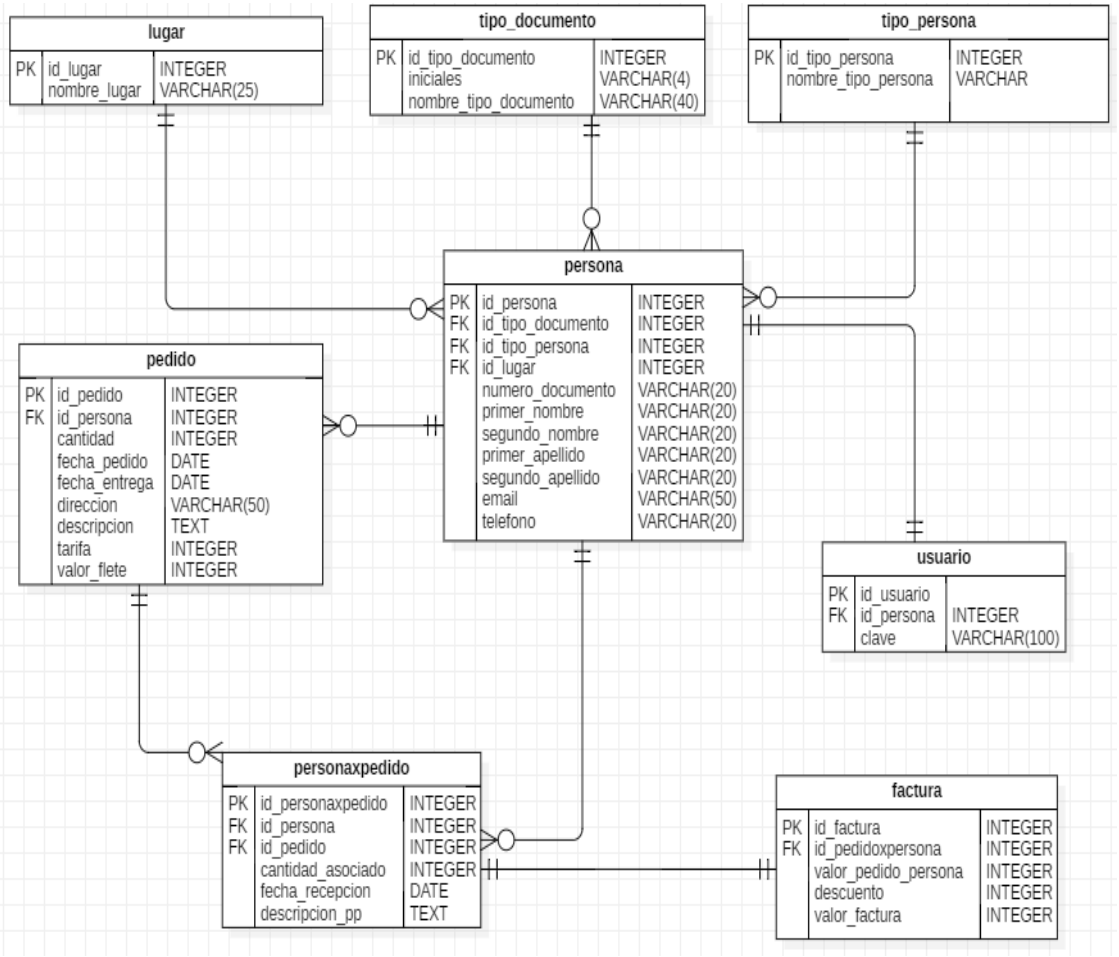


Figura 4. Modelado de la base de datos**.**

El diagrama de estructura del sistema permite observar la interacción del sistema, la manera en que funciona internamente el aplicativo para recibir peticiones y entregar información.

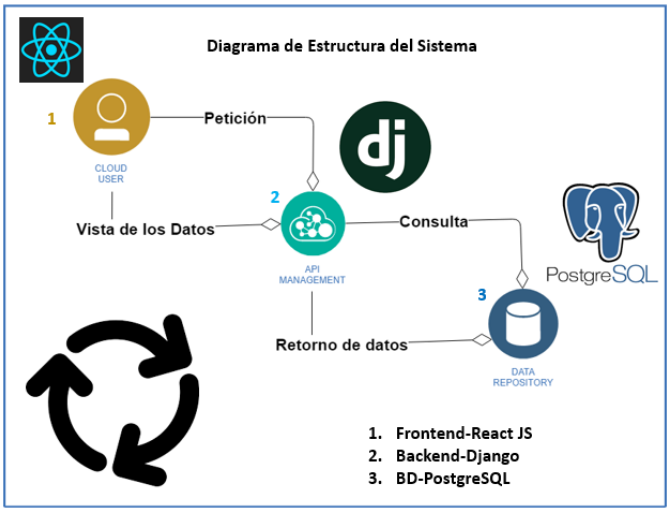


Figura 5. Flujo de navegación y estructura**.**

En cuanto al funcionamiento del proyecto se realizó un diagrama de flujo del sistema, el cual permite conocer la interacción de cada uno de los módulos.

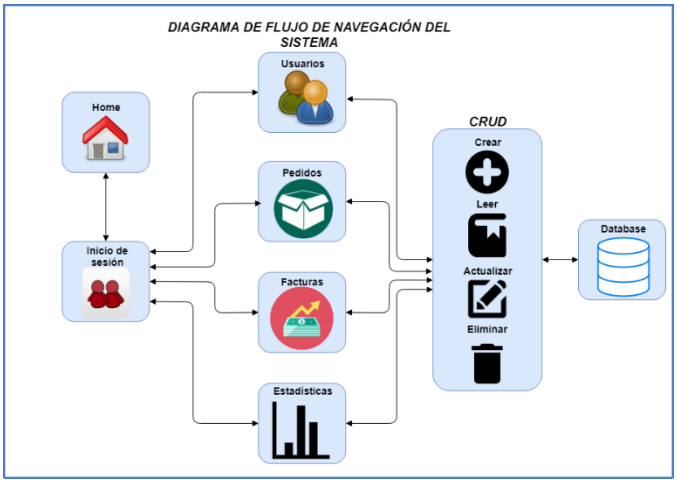


Figura 6. Diagrama de flujo de navegación del sistema.

Las herramientas utilizadas en dicho desarrollo fueron: como motor de base de datos PostgreSQL por su robustez y accesibilidad (PostgreSQL, 2019). ReactJS como librería basada en JavaScript para el desarrollo de aplicaciones web. En una encuesta acerca de la librería preferida, ReactJS obtuvo el segundo lugar von un 31.3% de preferencia. (Starck Overflow, 2019).

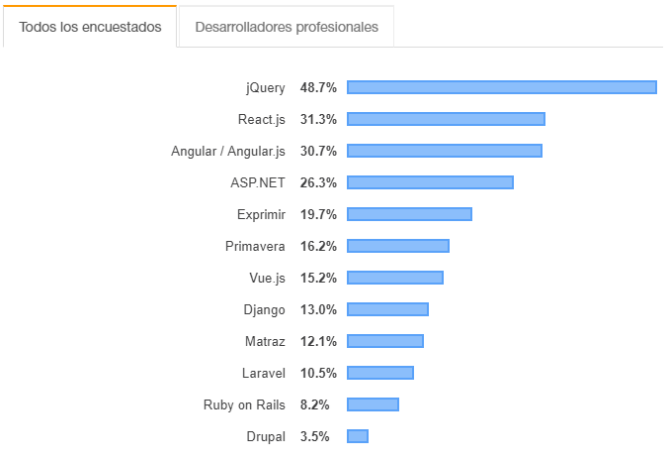


Figura 7. Resultado de librería preferida por desarrollares de aplicaciones web.

JavaScript como lenguaje de programación interpretado por su capacidad para ejecutar acciones del lado del cliente y diversidad de interacción con este, definido como un lenguaje de programación orientado a objetos (POO) (Alba, 2011). NodeJS es el entorno de ejecución multiplataforma escogido ya que el proyecto es realizado en JavaScript y funciona de manera asíncrona y orientada a eventos (NodeJS, 2019). Por su lado, React-Native se utilizó como framework para el desarrollo móvil ya que permite crear aplicaciones para los diferentes sistemas operativos de móviles (IOS y Android) y por su compatibilidad con JavaScript (Panigrahi, 2018). HTML 5 como lenguaje de etiqueta para construir páginas web con el que se moldea la forma en se estructura un sitio web (Gauchat, 2012). CSS3 abarca los estilos gráficos que se realizan en el diseño de aplicaciones web (el css3 va de la mano con el HTLM 5), y Boostrap 4 es el framework del CSS, conjunto de herramientas de código abierto (Boostrap, 2019). Por último, Visual Studio Code como la herramienta más importante donde se encuentran los editores de código que permite crear aplicaciones web empresariales.

3. Fase de Modelado:

Esta fase se desarrolló por medio de los casos de uso teniendo en cuenta los requisitos funcionales influyentes y relevantes para la funcionalidad final del aplicativo. Se realizó la construcción en el formato extendido para la descripción de los diferentes casos de uso; en las tablas se hace una descripción del caso de uso, la información que interviene y las condiciones para cada caso. En la tabla 1 se da un ejemplo de un modelo de caso de uso extendido en el menú principal del sistema.

Nombre Caso de Uso: Inicio de Sesión CU-01	
Actor (es):	Usuario, Sistema, Base de datos.
Descripción:	Es el formulario de inicio del sistema, en el cual se presenta la opción de inicio de sesión para usuarios.
Entradas:	Datos para inicio de sesión (correo electrónico y contraseña).
Salidas:	La confirmación e ingreso de sesión exitoso para el usuario.
Precondiciones:	El usuario debe estar creado en la base de datos y debe existir el formulario de inicio.
Postcondiciones:	El usuario tendrá la siguiente interfaz de su perfil.
Precedentes:	Ninguno.
Usan o extienden:	Crear Usuario CU-02, Mostrar usuario, modificar usuario.

Tabla 1. Caso de uso de inicio de sesión del sistema

Para la construcción y diseños de las vistas de la aplicación web y app móvil se elaboraron unos bocetos conocidos en el área de diseño como mockups, los cuales consisten en plasmar un modelo del posible diseño de la interfaz gráfica de usuario que dispondrá el cliente. En la Figura 8 se puede apreciar el diseño de la pantalla de ingreso al sistema en la interfaz del aplicativo web y en la Figura 9 la interfaz de la aplicación móvil.

Figura 8. Diseño de pantalla de ingreso en aplicación web**.**

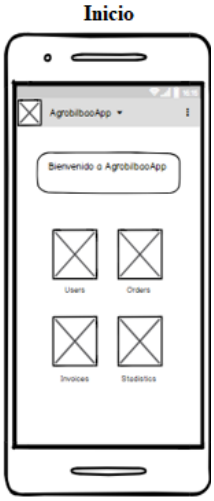


Figura 9. Diseño de pantalla de ingreso en aplicación móvil.

4. Fase de Construcción:

Para el sistema se utilizó Single-Page Application (SPA) como el patrón de diseño de software para este proyecto; el cual permite una mayor seguridad y escalabilidad del proyecto. Para la descripción del desarrollo y codificación del software tenemos cuatro secciones importantes, Frontend, Backend, Base de Datos y la App móvil las cuales serán relacionadas a continuación:

4.1 Fronted:

Incluye toda la lógica y construcción de los componentes para el despliegue de la información de manera gráfica, junto con la interacción hacia el cliente generando una agradable experiencia de usuario. Esta capa es ejecutada por el navegador web ya sea Google Chrome, Mozilla, Firefox, Opera, Safari u otro, y para su realización utilizamos como herramienta React JS.

En la Figura 10, se puede observar la codificación del método (saveData) encargado de registrar los usuarios, en el cual se captura la información que el usuario está digitando en el formulario, se asigna a una variable de tipo JSON (JavaScript Object Notation) y posteriormente realizamos una petición de tipo POST al backend de nuestra aplicación para culminar el proceso.

```
saveData() {
    if (this.form.current.reportValidity()) {
        this.setState({
            loading: true,
        });

        let postData = {
            numero_documento: this.form.current.numero_documento.value,
            primer_nombre: this.form.current.primer_nombre.value,
            segundo_nombre: this.form.current.segundo_nombre.value,
            primer_apellido: this.form.current.primer_apellido.value,
            segundo_apellido: this.form.current.segundo_apellido.value,
            email: this.form.current.email.value,
            telefono: this.form.current.telefono.value,
            id_tipo_documento: this.form.current.id_tipo_documento.value,
            id_tipo_persona: this.form.current.id_tipo_persona.value,
            id_lugar: this.form.current.id_lugar.value,
        };

        console.log(JSON.stringify(postData));

        axios.post(URL_API + "/person", postData, this.headers
        ).then(res => {
            if (res.status === 201) {
                // console.log("res: " + JSON.stringify(res.data));
                this.setState({
                    loading: false,
                });
                this.props.onDataSaved();
            }
        }).catch(error => {
            console.log(error);
        });
    }
}
```

Figura 10. Método para registrar usuario.

4.2 Backend:

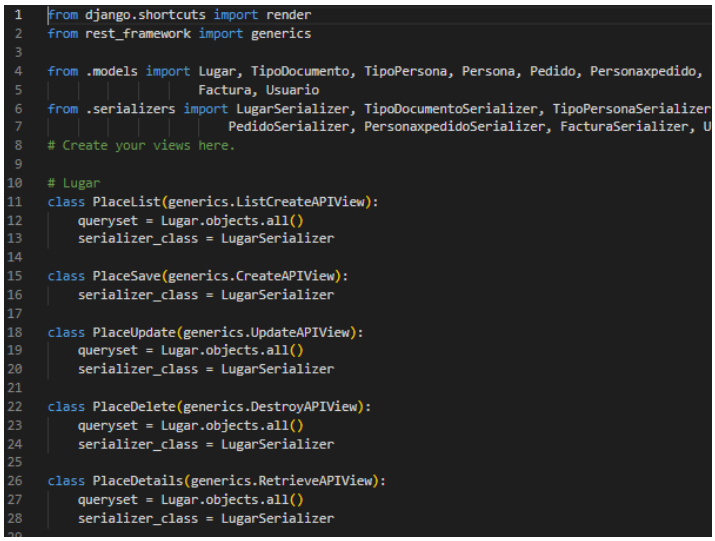
Esta capa se trabajó en el desarrollo de un API (Application Programming Interface), aplicación del lado del backend que va a tener una serie de rutas, métodos y lógica de control que permite hacer una interacción con la base de datos y ser consumida con el protocolo HTTP por cualquier tipo de cliente ya sea un cliente presente en un navegador web o desde una aplicación móvil.

Rutas: también conocidas como endpoints, son las URL que serán consumidas por el frontend. Estos endpoints serán el enlace de entrada de los datos para los distintos tipos de métodos HTTP a realizar. En la Figura 11 podemos observar un ejemplo de las rutas declaradas para los lugares.

```
from django.contrib import admin
from django.urls import path
from agro.queries import query
from agro.views import PlaceList, PlaceSave, PlaceUpdate, PlaceDelete, PlaceDetails, \...

urlpatterns = [
    path('admin/', admin.site.urls),
    # place
    path('api/places', PlaceList.as_view(),name='place_list'),
    path('api/place', PlaceSave.as_view(),name='place_save'),
    path('api/place/<pk>/update', PlaceUpdate.as_view(),name='place_update'),
    path('api/place/<pk>/delete', PlaceDelete.as_view(),name='place_delete'),
    path('api/place/<pk>', PlaceDetails.as_view(),name='place_details'),
    # TipoDocumento
    path('api/documenttypes', TipoDocumentoList.as_view(),name='documenttype_list'),
```

Figura 11. Ejemplo de construcción de endpoints del proyecto**.**

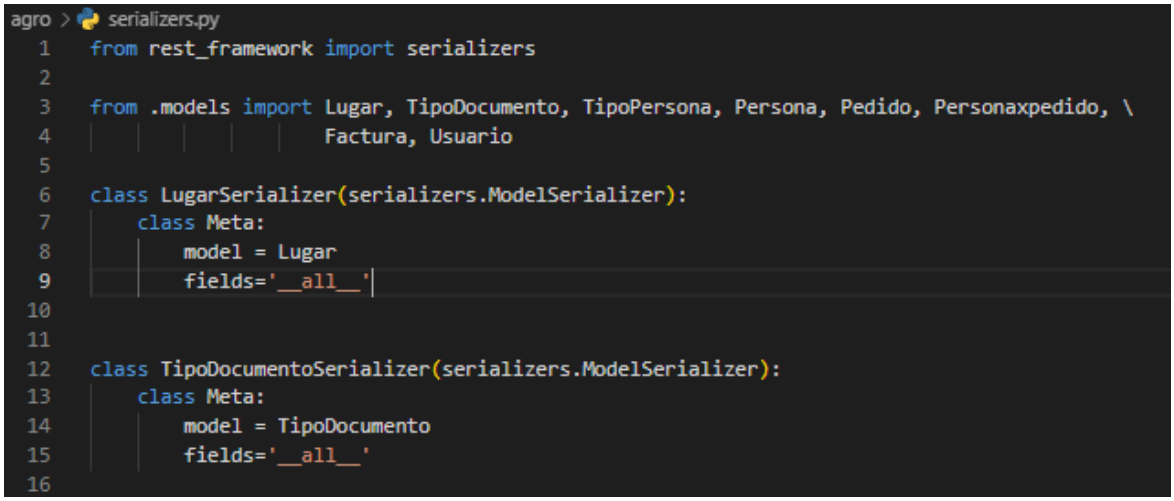
Vistas: estas ayudan a describir cada uno de los métodos respecto al serializador descrito, para ello estamos importando los serializadores creados por cada entidad involucrada como la Figura 12 lo muestra.



```python
from django.shortcuts import render
from rest_framework import generics

from .models import Lugar, TipoDocumento, TipoPersona, Persona, Pedido, Personaxpedido,
                    Factura, Usuario
from .serializers import LugarSerializer, TipoDocumentoSerializer, TipoPersonaSerializer
                         PedidoSerializer, PersonaxpedidoSerializer, FacturaSerializer, U
# Create your views here.

# Lugar
class PlaceList(generics.ListCreateAPIView):
    queryset = Lugar.objects.all()
    serializer_class = LugarSerializer

class PlaceSave(generics.CreateAPIView):
    serializer_class = LugarSerializer

class PlaceUpdate(generics.UpdateAPIView):
    queryset = Lugar.objects.all()
    serializer_class = LugarSerializer

class PlaceDelete(generics.DestroyAPIView):
    queryset = Lugar.objects.all()
    serializer_class = LugarSerializer

class PlaceDetails(generics.RetrieveAPIView):
    queryset = Lugar.objects.all()
    serializer_class = LugarSerializer
```

Figura 12. Ejemplo de construcción de vistas del proyecto.

Serializadores: permiten manejar los datos complejos como consultas o instancias que luego son convertidos para ser fácilmente transmitidos en formatos JSON o XML, en este desarrollo se utiliza el formato JSON puesto que su manejo es más sencillo, también nos evitan la reescritura de código. A continuación, la Figura 13 da un ejemplo de declaración de serializador para tipo de documento y para lugares:



agro > serializers.py

```python
from rest_framework import serializers

from .models import Lugar, TipoDocumento, TipoPersona, Persona, Pedido, Personaxpedido, \
                    Factura, Usuario

class LugarSerializer(serializers.ModelSerializer):
    class Meta:
        model = Lugar
        fields='__all__'


class TipoDocumentoSerializer(serializers.ModelSerializer):
    class Meta:
        model = TipoDocumento
        fields='__all__'

```

Figura 13. Ejemplo de construcción de serializador del proyecto.

Modelos: se declaran las clases con todas las tablas presentes en el modelo entidad relación de la base de datos. A continuación, la Figura 14 da un ejemplo de la construcción del modelo de la entidad de pedidos del proyecto.

agro > models.py

```python
# This is an auto-generated Django model module.
# You'll have to do the following manually to clean this up:
#   * Remove `managed = False` lines if you wish to allow Django to create, modify, and delete the table
# Feel free to rename the models, but don't rename db_table values or field names.
from django.db import models

class Pedido(models.Model):
    id_pedido = models.AutoField(primary_key=True)
    id_persona = models.ForeignKey('Persona', models.DO_NOTHING, db_column='id_persona', blank=True, null=True)
    cantidad = models.IntegerField(blank=True, null=True)
    fecha_pedido = models.DateField(blank=True, null=True)
    fecha_entrega = models.DateField(blank=True, null=True)
    direccion = models.CharField(max_length=50, blank=True, null=True)
    descripcion = models.TextField(blank=True, null=True)
    tarifa = models.IntegerField(blank=True, null=True)
    valor_flete = models.IntegerField(blank=True, null=True)

    class Meta:
        managed = False
        db_table = 'pedido'

```

Figura 14. Ejemplo de construcción de modelos del proyecto.

4.3 Base de datos:

Desde Django, se realizan las declaraciones de los parámetros necesarios para la conexión a la base de datos, en la Figura 15 se observar dicha configuración:

```python
# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'OPTIONS' : {
                'options': '-c search_path=produccion'
            },
        'NAME': 'agrobilbao',
        'HOST': '127.0.0.1',
        'USER':  'postgres',
        'PASSWORD': '*******',
        'PORT': 5432
    }
}
```

Figura 15. Parámetros de conexión a la base de datos.

Triggers: para el buen funcionamiento de nuestra aplicación web y móvil se implementó la utilización de disparadores (triggers) los cuales se pueden definir como procesos que se ejecutan

inmediatamente después que ocurre un evento en alguna tabla de la base de datos ya sea una inserción, eliminación o actualización.

APP MÓVIL

Dentro del proceso de codificación y desarrollo de la aplicación móvil se utilizó el framework ionic el cual soporta diferentes frameworks o librerías de desarrollo web. TypeScript es el lenguaje usado para la codificación de la aplicación el cual se transpila o transforma a JavaScript cuando vamos a compilar para algún sistema operativo ionic trabaja junto con la librería cordova para formar un proyecto de Android para Android studio, generando un archivo.

Figura 16. Codificación de la aplicación móvil.

5. Fase de Despliegue:

Esta fase se realizó de manera local en los computadores de trabajo, se deben tener en cuenta los siguientes puntos:

- Del lado del frontend de la app web, el despliegue se hace con NodeJS una herramienta que transforma el código en un archivo .html que el navegador entienda.
- Para el despliegue de lado del backend en Django se debe configurar un entorno virtual para Python el cual permite el despliegue del backend de manera óptima.
- Para la app móvil se hace la compilación desde Android y luego la subida al play store.

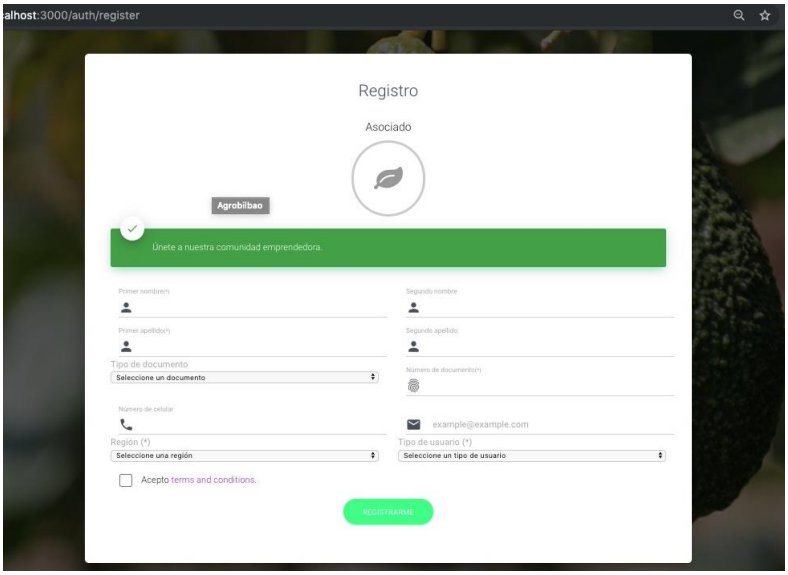


Figura 17. Ventana de registro de usuarios en aplicación web.

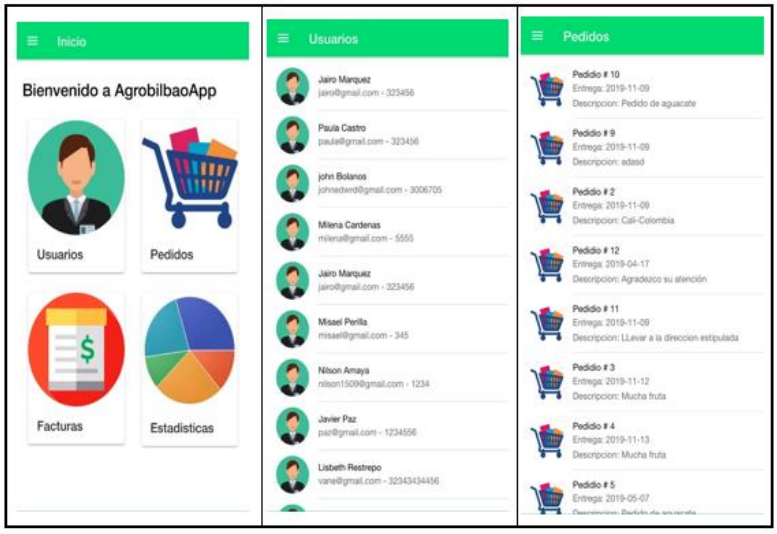


Figura 18. Ventanas en aplicación móvil.

RESULTADOS

Pruebas de caja Blanca:

Para este tipo de pruebas se utilizó la herramienta Apache JMeter en su versión 5.1.1 con la cual se hacen muchas peticiones para medir la carga que se esté soportando. A continuación, se van exponer los resultados obtenidos de las pruebas realizadas a la aplicación, las herramientas utilizadas, los métodos o procesos realizados para ello y la información utilizada para realizar dichas pruebas.

Escenario propuesto: Para el primer test, evaluando las posibles cargas que puede tener el aplicativo, las pruebas serán de consulta de información. Se hace una petición http de tipo GET, y para ello se siguen los siguientes pasos de configuración:

- Se especifica la ruta a la cual se le hace la petición: 192.168.0.11 (en este caso)
- Se selecciona el método, para este caso es de tipo: GET
- En el apartado de "Path" se colocan las extensiones de la ruta y que van a intervenir en la petición: api/orders.
- Se especifica el puerto donde se encuentra desplegado: 8000

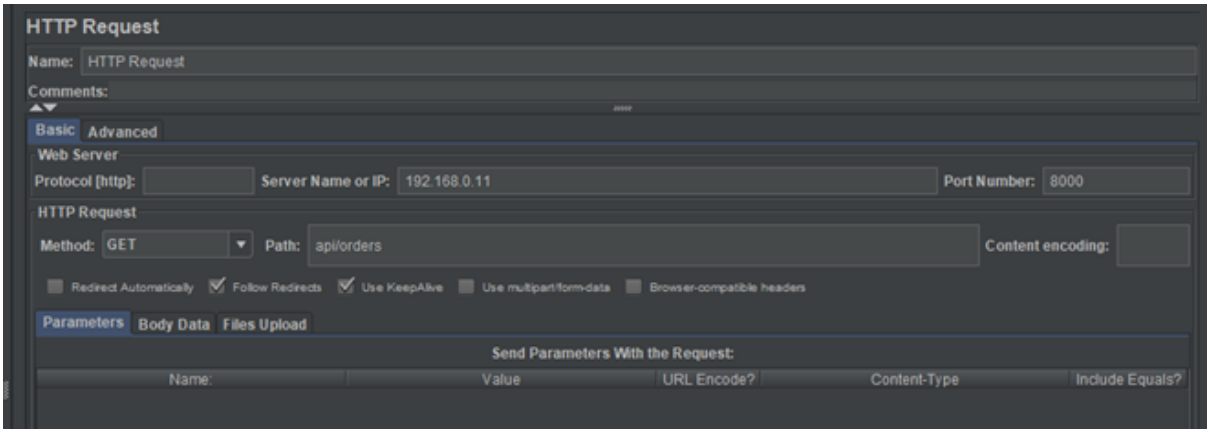


Figura 19. Configuración de Jmeter para petición GET

Para la prueba usamos un número de 100 usuarios por segundo, una carga de peticiones la cual el tiempo de la simulación duró 5 segundos y se obtuvieron los siguientes resultados:

Se obtuvo un resultado de un 94% de efectividad de las peticiones hechas, lo cual es un porcentaje importante para una carga de 100 usuarios en un segundo. En la figura 38 podemos observar el resultado de las peticiones hechas donde podemos ver la data de respuesta y en la figura 39 se observa el error obtenido, la latencia, la cantidad de bytes de información obtenida y el tiempo de respuesta.

DISCUSIÓN

Con la implementación de este sistema de información en empresas y asociaciones del gremio agricultor tienen más posibilidades de mejora en los resultados de sus procesos internos y externos, el software de este proyecto tiene características que permiten una adecuada escalabilidad; que dan la posibilidad de agregar nuevos módulos o funcionalidades que mejoren el funcionamiento del

sistema y así darle el mejor beneficio posible al campesino, promoviendo la actividad agrícola en la región, lo cual beneficia la economía de este entorno.

Incentivar al campesino agricultor para que se adapte al uso del software, promoviendo una cultura de uso de tecnologías que ayuden a mejorar el ejercicio agrícola que traerá beneficios como la disminución del desplazamiento hacia las ciudades por falta de oportunidades, el uso de la tecnología puede disminuir la brecha que existe entre el campesino productor y el cliente o consumidor final; eliminando intermediarios para que el campesino aumente su rentabilidad.

Para un uso adecuado del sistema de información se recomienda realizar una buena capacitación de uso de los aplicativos al personal de la asociación tanto en la aplicación web como en la móvil; esto debido a que la población campesina no está muy familiarizada con el uso de las nuevas tecnologías.

CONCLUSIONES

El desarrollo de este proyecto permitió mejorar habilidades de investigación, análisis y estudio, para conocer lo variable y cambiante que es el mundo del desarrollo de software. Para lo cual se necesita estar de manera constante actualizando conocimientos, conocer las nuevas tecnologías que ayudan en el día a día en estas carreras como profesionales.

Se logró conocer el ejercicio agrícola de una parte del país, conociendo las actividades que realiza un agricultor en su diario vivir, para producir en este caso el aguacate. El resultado fue entender que con tecnología y con la ayuda del software se puede ayudar a mejorar procesos en el campo; el potencial para implementar sistemas de información en el sector agrícola de Colombia es de grandes dimensiones y es algo que debe explotarse en el futuro para el beneficio de todos los actores.

La asociación AGROBILBAO ya contaba con estrategias en las áreas de producción, comercialización y exportación del aguate Hass y el sistema de información desarrollado para esta organización permitió al usuario administrador de prueba, por la estructura de su diseño, una comunicación más rápida y organizada de la información, tales como el acceso a los datos de los clientes de la asociación, los pedidos del producto y estadísticas para conocer cuánto progreso han tenido las ventas.

Por medio de las pruebas de funcionamiento realizadas por el equipo de trabajo se pudo comprobar su correcta funcionalidad y percepción de fallas en el desarrollo para su posterior corrección.

Referencias

Suarez, Viviana. 2019, Abril 17. ***80 % de los cultivos de aguacate hass son de campesinos.*** **[En línea]. Disponible en: https://www.elcolombiano.com/negocios/economia/80-de-los-cultivos-de-aguacate-hass-son-de-campesinos-IB10557359**

Ministerio de Agricultura. 2018, Dic 12. ***Acerca de Agronet*****. [En línea]. Disponible en: http://www.agronet.gov.co/agronet/Paginas/default.aspx**

Joyanes, J. F. 2014, Junio. ***Sistemas de Información en El Sector Agrícola*****. [En línea]. Disponible en: es.scribd.com/document/228070051/Sistemas-de-Informacion-en-El-Sector-Agricola**

AgroMovil. 2014. ***Innovación Tecnológica.*** **[En línea]. Disponible en: https://www.redinnovagro.in/docs/agromovil.pdf**

PostgreSQL. 2019, Abril 15. ***Acerca de postgresql*****. [En línea]. Disponible en: https://www.postgresql.org/about/**

Stack Overflow, 2019. ***Resultados de Framework y librería de preferencia*****. [En línea]. Disponible en: https://insights.stackoverflow.com/survey/2019#overview**

Alba, P. R. 2011. ***MANUAL DE JAVASCRIPT. Madrid: EDITORIAL CEP S.L.***

NodeJS. 2019, Marzo 26. ***Acerca de Node.js.*** **[En línea]. Disponible en: https://nodejs.org/es/about/**

Panigrahi, Kiran. 2018. ***React Native*****. Tutorials Point.**

Gauchat, J. D. 2012. ***El gran libro de HTML5, CSS3 y Javascript. Barcelona: MARCOMBO, S.A. 2012.***

Bootstrap. 2019, Marzo 7. ***Bootstrap.*** **[En línea]. Disponible en: https://getbootstrap.com/**