UNIVERSIDAD TECNOLÓGICA CENTROAMERICANA

FACULTAD DE INGENIERÍA

PRÁCTICA PROFESIONAL

LAUREATE INTERNATIONAL UNIVERSITIES

PREVIO A LA OBTENCIÓN DEL TÍTULO

INGENIERO EN SISTEMAS COMPUTACIONALES

PRESENTADO POR:

11441181 DENNIS JOEL CÁRCAMO ELVIR

ASESOR: LIC. TANIA LUCILA MEZA AMADOR

CAMPUS TEGUCIGALPA; ENERO, 2021

AGRADECIMIENTOS

Se agradece a la Universidad Tecnológica Centroamericana por brindar el soporte y asesoría necesaria en la ejecución de la práctica profesional así mismo se agradece a Laureate International Universities por brindar un espacio de trabajo para el practicante Dennis Joel Cárcamo Elvir en su departamento de IT en donde pueda llevar a cabo su práctica profesional.

Dedico este logro primeramente a Dios a quien le debo mi vida y las oportunidades que me ha brindado, a mis amigos y compañeros que me ayudaron en momentos difíciles y de confusión, a Rick y Susan Bledsoe que me aceptaron como parte de su familia y por confiar en mí, a Lydia y Dennis Whitehead, a Mike y Marty Edwards, a Katie y Todd Moddy y a Greg y Eva Vaughn por ser parte de mi larga formación bajo la guía de Dios, por apoyarme económicamente y por permitirme demostrarles mis deseos de superación.

RESUMEN EJECUTIVO

El presente documento es un informe que se presenta como requisito para optar al título de Ingeniero en Sistemas Computacionales, describe las actividades realizadas en el tiempo que dura la práctica profesional que comprende el periodo de julio a diciembre de 2020, la cual se realizó en el departamento de IT Operations de Laureate International Universities bajo el cargo de desarrollador de software.

Laureate International Universities posee una oficina en Honduras, la cual consta de un departamento de IT, este departamento es el encargado de todas las gestiones tecnológicas que Laureate Honduras pueda tener, es también el encargado de administrar los bienes físicos como computadoras y enrutadores hasta activos de software como licencias y programas.

En el año 2018 se crea la idea de tener una plataforma que integre diferentes módulos y características que faciliten la administración de los activos, que agilicen todas estas gestiones y que se puedan generar reportes de los activos físicos y de software, esto para evitar seguir haciendo estas operaciones de forma manual e ineficiente.

Con la importancia en mente de mejorar los tiempos de estos procedimientos y facilitar la gestión de las operaciones nace con el nombre de InvIT (Inventory IT) la plataforma en la que las primeras características surgieron y que ha tenido un continuo crecimiento desde entonces.

Actualmente la plataforma de InvIT ya integra algunos de los sistemas que se utilizan con mayor frecuencia, pero la actualización, mejora y mantenimiento de este se ha visto afectado por la falta de tener un desarrollador de software que realice estas tareas.

A medida que las operaciones del departamento de IT se vuelven más complejas la necesidad de poder integrar en un solo sistema todos los servicios que brinda a sus colaboradores, este sistema se ve en la obligación de continuar creciendo en características para que posea una funcionalidad total que abarque los aspectos que al departamento de IT Operations Interesa.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

LISTA DE SIGLAS Y GLOSARIOS

API	Interfaz de Programación de Aplicaciones.
CRAI	Centro de recursos para el aprendizaje y la investigación.
DB	Base de datos.
DB-API	Interfaz de Aplicación Programable para Base de Datos.
ERD	Diagrama Entidad Relación.
HTML	Lenguaje de Marcas de Hipertexto.
HTTP	Protocolo de Transferencia de Hipertexto.
InvIT	Inventario de IT.
IT	Tecnologías de Información.
JSON	Objeto de Notación de JavaScript.
MVC	Patrón Modelo Vista Controlador.
ORM	Mapeador de Objetos Relacionales.
PDF	Formato de Documento Portátil.
QA	Control de calidad.
SQL	Lenguaje de Consultas Estructuradas.
TIC	Tecnologías de Información y Comunicación.
URL	Localizador de Recursos Uniforme.

VPN	Red Privada Virtual.
WSGI	Web Server Gateway Interface.
XLSX	Documentos de salida de hojas de cálculo de Microsoft Excel.

Active Directory	Es una estructura en forma de jerarquía que guarda toda la información de objetos dentro de una red. *(iainfoulds, s. f.)*
Backend	Son los procesos del lado del servidor. (*¿Qué es BACKEND y FRONTEND?*, s. f.)
Framework	Es una estructura conceptual y de tecnologías con soporte definido, con artefactos y modelos de programación específicos. *(¿Qué es un framework y para qué se utiliza?, s. f.)*
Frontend	Es el cliente y todo el resultado que se muestran al usuario. (*¿Qué es BACKEND y FRONTEND?*, s. f.)
JavaScript	Lenguaje de programación ligero e interpretable, es un lenguaje de comandos para páginas web. (*JavaScript*, s. f.)
Librería	Conjunto de archivos que se usan para facilitar la programación. (*Librerías para la programación web*, s. f.)
Sintaxis	Conjunto de reglas y formas que debe tener un lenguaje para su correcta interpretación. *(Sintaxis de Programación, 2016)*

Web Server Gateway Interface	Es una especificación en la que se detalla cómo un servidor web se comunica con una aplicación para procesar las solicitudes *(Nacho Alonzo ¿Qué es un WSGI?, s/f)*

I. INTRODUCCIÓN

Desde el auge de las tecnologías de la información se ha visto un crecimiento de la cantidad de compañías que se dedican al rubro de la producción de software y tecnologías de información, así como las compañías que producen elementos de limpieza poseen un departamento de producción de la misma forma las compañías actuales poseen un departamento de Tecnologías de Información (IT).

Muchas de las operaciones que se realizan a diario dentro de una empresa se hacen actualmente utilizando algún tipo de software o sistema informático, es responsabilidad del departamento de IT atender todas las necesidades de cómputo, asesorar a los usuarios en el uso del software, también es responsable de la configuración, asignación y mantenimiento del equipo de cómputo, conexión a internet y acceso a las telecomunicaciones para asegurar la estabilidad y seguridad dentro de los sistemas informáticos de la compañía.

El departamento de IT Operations es el ente encargado de las operaciones mencionadas anteriormente. Una de las operaciones que mayor trabajo, atención a los detalles y complejidad tiene es la administración del Inventario de Laureate, este inventario incluye activos tangibles como el equipo de cómputo y activos no tangibles como la información de los colaboradores de Laureate y software.

Es con el fin de mejorar la forma en que IT Operations realiza sus tareas que nace la necesidad de desarrollar la plataforma de uso exclusivo para el equipo llamada InvIT, este sistema promete mejorar el tiempo en que cada una de las gestiones de Inventario se realizan, promete unir funciones externas que Integren y automaticen tareas como la generación de reportes, envío de notificaciones a los colaboradores y a los administradores del sistema entre otras, de esta forma el equipo de IT Operations podrá realizar sus tareas de manera eficiente con la ayuda de un sistema de administración de inventario completamente funcional.

El presente documento mostrará las generalidades de Laureate International Universities, se describe el puesto en el que el practicante trabajará, cuáles serán sus objetivos generales y específicos en los que se estará desenvolviendo.

En el marco teórico se mostrarán las tecnologías que se están utilizando en la versión actual de InvIT y se describirán las tecnologías que se estarán implementando para el desarrollo de

los nuevos módulos, también se mostrará la arquitectura utilizada en el sistema actualmente para dar al lector una visión clara de cómo se integran todas estas tecnologías.

En la sección de desarrollo se describirán detalladamente cada una de las actividades a realizar durante la práctica profesional para obtener una mejor comprensión de los objetivos asignados.

II. GENERALIDADES DE LA EMPRESA

2.1 DESCRIPCIÓN DE LA EMPRESA

Laureate cree en el poder de la educación para cambiar vidas. Laureate Education, Inc. es un proveedor líder de educación superior de calidad a través de su red global de instituciones de educación superior otorgadoras de títulos y de sus relaciones colaborativas de vanguardia con empresas líderes de la industria *(Sobre Laureate, s/f)*.

Laureate se enorgullece de ser parte de un movimiento global dedicado a hacer del mundo un lugar mejor. Estamos comprometidos con tener propósito en todo lo que hacemos, y de transmitir conocimientos y habilidades que ayuden a nuestros estudiantes a convertirse en agentes de cambio y aprendices de por vida. De esta manera, somos Here for Good. Ya sea a través de nuestra red de clínicas de salud dirigidas por estudiantes que ofrecen servicios gratuitos o de bajo costo a más de 150,000 pacientes cada año, o con nuestros ganadores del premio Here for Good que han recibido reconocimiento mundial por su excepcional impacto social, nuestras instituciones, estudiantes, personal administrativo y docentes juegan un papel importante en cambiar el mundo *(Sobre Laureate, s/f)*.

Alineados a nuestro compromiso de ser Here for Good, estamos orgullosos de ser una de las corporaciones B Certificadas® y Corporaciones de Beneficios Públicas (PBC) más grandes del mundo. Además, nuestra PBC es la empresa más grande de este tipo que se cotiza públicamente en cualquier bolsa de valores del mundo. Ser una PBC y una B Corp nos permite integrar legalmente nuestra misión en todo lo que hacemos, lo que nos empuja a encontrar formas significativas de mejorar nuestro desempeño social y ambiental. Nos ayuda a equilibrar nuestro propósito y beneficio, cerrando la brecha entre nuestra comunidad académica y nuestras operaciones comerciales. Nos esforzamos por ser no solo los mejores del mundo, sino también los mejores para el mundo. Sin embargo, lo que nos diferencia es la escala y la plataforma que tenemos para equipar a nuestros estudiantes a nivel mundial con las habilidades y la conciencia para ser embajadores en este movimiento, preparándolos para liderar empresas y comunidades innovadoras que están decididas a efectuar un cambio global *(Sobre Laureate, s/f)*.

2.2 DESCRIPCIÓN DEL DEPARTAMENTO

En el departamento de tecnología de información existe la división de IT Operations encargada de todas las gestiones de tecnología de la compañía, se encarga de la administración de todos los sistemas de Información de la empresa y las operaciones de soporte de tipo tecnológico.

2.3 OBJETIVOS DEL PUESTO

2.3.1 OBJETIVO GENERAL

Actualizar y desarrollar los módulos de administración de inventario de IT (InvIT) y control del AssetExplorer de la empresa.

2.3.2 OBJETIVOS ESPECÍFICOS

- Creación de módulo de reportería para la aplicación de Inventario.
- Automatización y configuración de los reportes para el envío en formato PDF/XLS.
- Mejoras en la seguridad del sistema InvIT, a nivel de front-end, red y autenticación.
- Creación del módulo de notificaciones para la aplicación del Inventario
- Crear un control de cambios y movimientos en el inventario-tabla y reportes de LOGS.

III. MARCO TEÓRICO

En esta sección se muestra de forma detallada las principales funciones que InvIT debe cumplir y qué tecnologías este implementa, también se describen las herramientas y términos relacionados al sistema, se explica esto para proveer al lector una mejor idea de la funcionalidad del sistema y cómo ha ido evolucionando su desarrollo.

El inventario generalmente es el activo más importante de toda la organización, el inventario busca mejorar todas las operaciones de la empresa por lo que llegan a convertirse en una necesidad absoluta porque es el medio que permite a la empresa desarrollarse y crecer. *(Bayas & Martínez, 2017)*

La correcta administración del inventario evitará a la compañía enfrentarse a problemas financieros, es uno de los elementos fundamentales de la empresa porque es el activo que genera mayor rentabilidad, es también el que mueve a la organización porque se encarga de mantener en producción a la organización para que se pueda mantener la demanda. *(Durán Yosmary, 2012.p.56)*

La gestión de inventario se define como la correcta administración del registro, salidas y compras dentro de una compañía, existen también tareas implícitas como la generación de reportes de inventario y la organización.

En la antigüedad estas tareas se llevaban en forma física lo que conlleva muchas desventajas, para consultar el inventario o tratar de encontrar un elemento específico tomaba demasiado tiempo y a veces era casi imposible consultarlo cuando el inventario era demasiado grande.

Muller, M. 2005 menciona los puntos que se deben tomar en consideración si se desea llevar una administración de inventario exitosa, algunas de estas razones son:

- Capacidad de predicción: el inventario debe mantener un equilibrio entre lo que se necesita y en lo que se usa.
- Estabilidad de suministros: asegura tener materia con la que trabajar.

Correa, A. (2010) menciona que actualmente las tecnologías de información y comunicación (TIC), son el medio por el cual las empresas se valen para multiplicar su productividad y eficacia en el manejo del inventario por lo que muchas empresas consideran las TIC como herramienta esencial en el manejo del Inventario.

Laureate International Universities no es la excepción, existen en su departamento de IT Operations sistemas informáticos que ayudan a la gestión de inventario. Actualmente se utiliza el software llamado AssetExplorer en el que existe un registro para cada uno de los elementos físicos dentro de la compañía, desde mobiliario, equipo de cómputo y de red, hasta mochilas y equipos de sonido. También es responsabilidad del departamento de IT Operations mantener la información de sus colaboradores y las relaciones que estos tienen con el inventario.

Es por esta relación de colaborador – inventario que nace InvIT la plataforma que se empezó a desarrollar con el objetivo de hacer la administración de inventario más fácil y que se mantenga en conformidad para que cumpla con los estándares de la auditoría. InvIT fusiona dos bases de datos para su completa funcionalidad, estas bases de datos son:

- assetexplorer.
- invitdb.

Ambas bases de datos están en PostgreSQL.

Manage Engine AssetExplorer es una aplicación web que ayuda a los directores de IT a monitorear y a gestionar los activos dentro de la red de la compañía desde el momento de la planeación de obtener nuevos elementos de inventario hasta la fase final del elemento, esta poderosa herramienta cuenta con varias formas de descubrir y gestionar los equipos dentro de la red. Es capaz de administrar recursos de hardware y de software. *(Lau, s. f.)*

Algunas de las funcionalidades más importantes son:

- Monitoreo del ciclo de vida de todos los activos.
- Ver toda la información de los activos y sus relaciones.
- Generación de reportes.

En la Ilustración 1 se puede observar el diagrama de aplicación de Asset Management. Esta muestra las integraciones a la aplicación AssetExplorer que se le han hecho y las bases de datos que se utilizan.

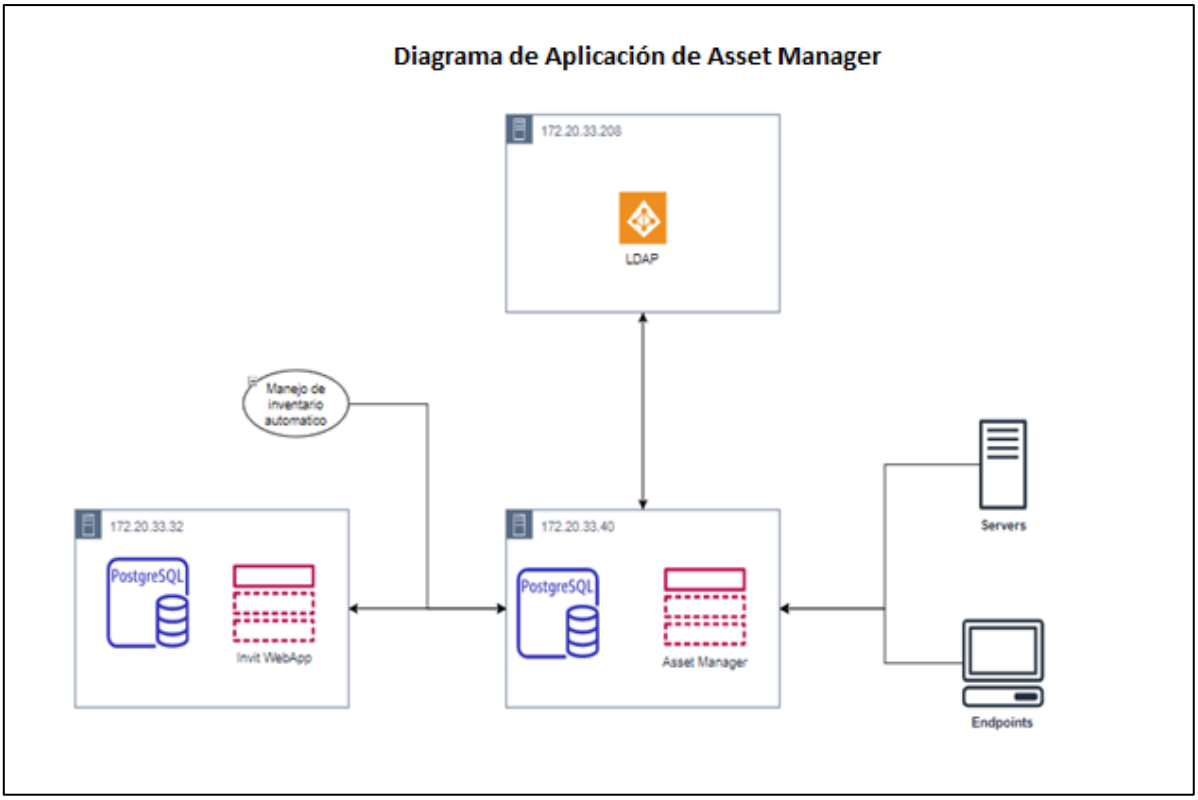


Ilustración 1 - Diagrama de aplicación de Asset

Fuente*: (Arquitectura de Asset Management, 2019)*

El sistema llamado "InvIT" (Inventory IT), consiste en una plataforma web para la gestión de hojas de inventario para un departamento de Laureate International Universities en específico. Su función general es mantener un eficiente control interno de los diferentes activos que maneja la empresa y de los usuarios responsables de dichos productos. De igual forma generar estadísticas que faciliten la toma de decisiones diarias en el departamento de IT Operations.

Actualmente InvIT posee:

- Un módulo web para la creación y modificación de Hojas de inventario.
- Actualización e integración de la aplicación de AssetExplorer.
- Un módulo web para la administración de hojas de inventario en formato pdf.
- Un Tablero para mostrar estadísticas a tiempo real de una manera amigable y fácil de entender.
- Un módulo para la generación de tablas y reportes.

El aplicativo va dirigido exclusivamente al personal administrativo del departamento de IT Operations de Laureate, existe la escalabilidad dentro del mismo para crear roles y controlar el contenido disponible para cada rol, esas decisiones quedarán bajo la responsabilidad del administrador.

El proyecto InvIT es original de la empresa Laureate International Universities, desarrollado para que funcione como una extensión personalizada de la aplicación AssetExplorer, se conecta a la misma base de datos y mantiene una sincronía entre las dos aplicaciones haciendo el manejo del inventario más fácil y amigable al usuario. En la Ilustración 2 se puede ver la arquitectura actual de InvIT.

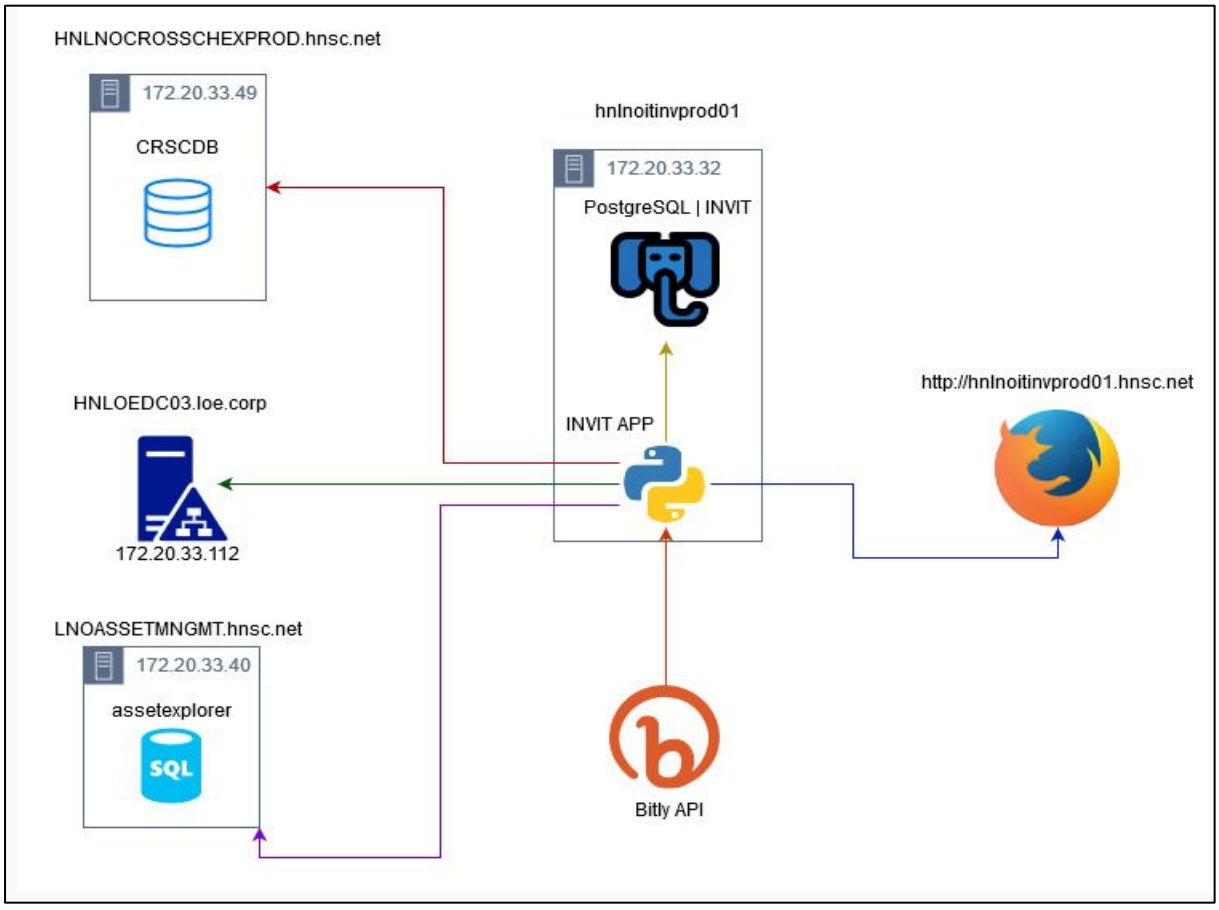


Ilustración 2 - Diagrama de arquitectura de software de InvIT

Fuente: *(Arquitectura de InvIT, 2019)*

Hasta hace algunos años se conoce al backend como la forma en la que el servidor renderizaba, procesaba y respondía a las peticiones de contenido provenientes de una interfaz gráfica.

Cuando una página se procesa del lado del servidor todos los procesos que conlleva la renderización de una página HTML y el procesamiento de sus datos interpretados por el navegador se llevan a cabo en un servidor remoto. Algunos de estos procesos incluyen cualquier lógica de procesamiento que tiene la aplicación e incluso la consulta de información a la base de datos.

Todos estos procesos no son visibles para el usuario porque están por detrás (back).

Del otro lado se tiene el frontend, este se conoce como los elementos renderizados del lado del cliente o lo que el cliente puede ver. En términos prácticos se le conoce como lógica de presentación a todo lo que el navegador puede interpretar y mostrar al usuario. *(Pastronio, 2019)*

El frontend de InvIT está desarrollado con las siguientes tecnologías y herramientas:

- Angular 5: es un framework pensado para desarrollar aplicaciones de una sola página (SPA). Su estructura base está desarrollada en JavaScript y lenguaje de marcado de plantillas como HTML.

  Con angular el desarrollo es más rápido pues contiene en su estructura herramientas y partes que se han creado con anticipación listas para su uso, esto se traduce a un lanzamiento más rápido y fácil de la aplicación.

  Angular utiliza el diseño Modelo Vista Controlador (MVC) para el desarrollo de aplicativos de internet. Mejora todas las tareas del lenguaje haciendo que las tareas difíciles del mismo sean más sencillas, este uso ayuda a recuperar, insertar y modificar datos sin importar el tipo de interfaz del usuario. *(Kumar, 2018.p.1)*

  En la Ilustración 3 se puede ver la estructura por defecto que angular crea al momento de empezar un nuevo proyecto utilizando el comando *"ng new nuevoproyecto-app"*.

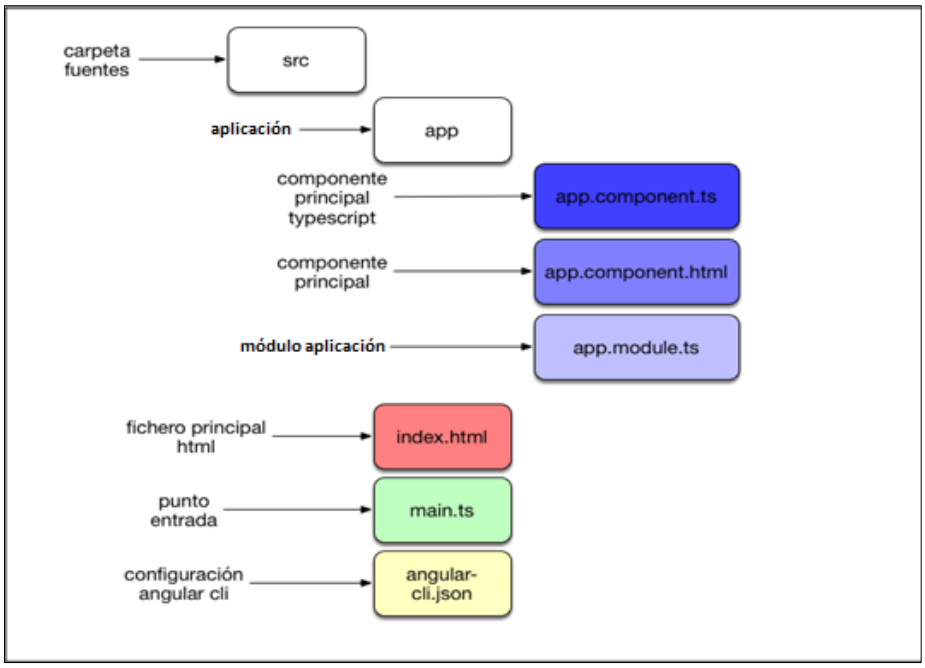


Ilustración 3 - Estructura de un proyecto en Angular

Fuente*: (Caules, 2018)*

Angular se caracteriza porque cada uno de sus componentes están divididos en 4 archivos y para ejemplificar esto de la mejor forma vea el ejemplo de un proyecto llamado "Myproject" que tendrá los siguientes archivos:

- Myproject.component.css: este archivo tiene los estilos del componente.
- Myproject.componente.html: este archivo define la estructura del componente.
- Myproject.component.ts: este archivo define la funcionalidad del componente.
- Myproject.module.ts: este archivo define el módulo al que pertenece el archivo.*(Caules, 2018)*

- Bootstrap 4.0: es el kit de herramientas de código abierto para desarrollo de front-end más conocido alrededor del mundo. Sirve para diseñar y personalizar los sitios web y móviles, esta librería incluye variables, sistema de cuadrícula receptivo y muchísimos componentes prediseñados y potentes complementos de JavaScript. *(Otto & Thornton, s. f.)*

Bootstrap se especializa en darle estilo al lenguaje de marcado, con un simple campo en los parámetros de las etiquetas estas toman un estilo previamente creado y personalizado

que facilita la creación de interfaces de usuario. En la Ilustración 4 se puede ver que el estilo cambia con agregar el campo *"class"* con un valor predefinido por Bootstrap.

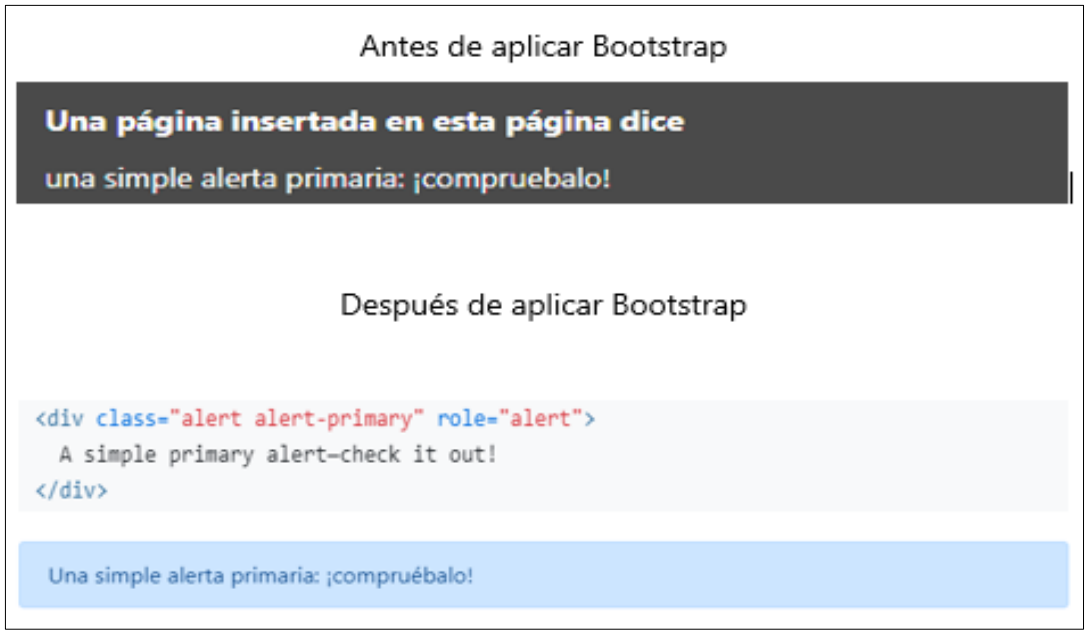


Ilustración 4 - Alerta antes y después de aplicar Bootstrap

Fuente*: (Elaboración propia)*

- FontAwesome 4.7: es una librería de iconos que se puede usar en cualquier lado que se está utilizando un prefijo de estilo y agregando el nombre del icono que se desea. Con FontAwesome simplemente se necesita una etiqueta HTML para el uso de los iconos, se puede usar la etiqueta por ser la más breve, la etiqueta por ser la más usada o que es semánticamente más correcta.*(Font Awesome, s. f.)*

En la Ilustración 5 se puede ver un ejemplo de cómo usar uno de los iconos de esta librería.

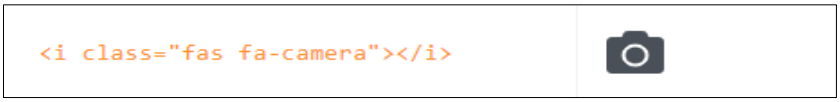


**Ilustración 5 - Utilizando FontAwesome en la etiqueta **

Fuente*: (Elaboración propia)*

- Chart.js: librería de JavaScript para la animación de gráficos. Esta librería es de código abierto mantenido por la comunidad, permite la visualización de la información en 8 diferentes formas, trabaja excelente con lienzos de HTML. *(Chart.js | Gráficos HTML5 de código abierto para su sitio web, s. f.)*

El backend de InvIT está desarrollado con las siguientes tecnologías y herramientas:

- Flask Python: es un framework WSGI ligero. Está pensado para que el lanzamiento de las aplicaciones sea rápido y sencillo siempre manteniendo la capacidad de crecer en el futuro y convertirse en una aplicación más robusta. Su origen se da como una envoltura a Werkzeug y Jinja. *(Ronacher, s. f.)*

  La instalación de Flask se puede hacer utilizando el comando "*pip install -u flask*". En la Ilustración 6 se puede ver un ejemplo sencillo de cómo crear una aplicación flask y como ejecutarla con el comando "*$ env Flask_App = hello.py flask run*", donde env es el ambiente virtual y "hello.py" es el nombre de la aplicación.

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World!"
```

```
$ env FLASK_APP=hello.py flask run
 * Serving Flask app "hello"
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Ilustración 6 - Aplicación utilizando Flask

Fuente*: (Ronacher, s. f.)*

- API Flask-RESTful: es una extensión para Flask que agrega soporte para la creación de APIS que se comunican a través del protocolo HTTPS. Flask ayuda a la creación de Interfaces de Aplicaciones Programables (API) a través de los modelos ORM. *(Quickstart — Flask-RESTful 0.3.8 documentation, s. f.)*

En la Ilustración 7 se puede ver una API sencilla creada con Restfull. Existe una clase llamada *"HelloWorld"* y dentro de esta una función llamada *"get"* que pertenece a los verbos de llamado de los métodos de petición HTTP.

```python
from flask import Flask
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

class HelloWorld(Resource):
    def get(self):
        return {'hello': 'world'}

api.add_resource(HelloWorld, '/')

if __name__ == '__main__':
    app.run(debug=True)
```

Ilustración 7 - API con Flask-RESTFul

Fuente*: (Quickstart — Flask-RESTful 0.3.8 documentation, s. f.)*

Flask RESTFul es capaz de entender y trabajar con varios tipos de retorno de los métodos de vista. Así como en Flask, puede devolver cualquier valor iterable que se convertirá en una respuesta e incluso un objeto de respuesta Flask. *(Quickstart — Flask-RESTful 0.3.8 documentation, s. f.)*

- SQLAlchemy: es un paquete de herramientas para Python SQL y un mapeador de objetos relacionales (ORM) que brinda a los desarrolladores toda la potencia y la flexibilidad del lenguaje SQL. SQLAlchemy consta de dos partes principales el núcleo y el ORM, el núcleo o core ya es un kit de herramientas en sí mismo que proporciona toda la abstracción sobre las múltiples aplicaciones y comportamiento de una Interfaz de Aplicación Programable para Base de Datos (DB-API), así como un lenguaje de expresiones SQL que permite usar todo el lenguaje de consultas a través de sentencias de Python. *(Características - SQLAlchemy, s. f.)*

El ORM estandariza en un sistema de configuraciones declarativas que permiten la creación de clases tomando en consideración la metadata de las tablas y la definición de la clase del usuario. Utilizando la función "*mapper()*" cualquier función en Python se puede asociar a

una vista o tabla de la base de datos. En la Ilustración 8 se pueden ver las sentencias de SQL tradicional para solicitar información.

```
import sqlite3
conn = sqlite3.connect('example.db')

c = conn.cursor()
c.execute('SELECT * FROM person')
print c.fetchall()
c.execute('SELECT * FROM address')
print c.fetchall()
conn.close()
```

Ilustración 8 - Forma tradicional de solicitar datos por SQL

Fuente*: (Introductory Tutorial of Python's SQLAlchemy, 2013)*

En la Ilustración 9 se puede ver la forma en la que SQLAlchemy hace el mapeo de las tablas a una clase de Python en la que incluye toda la información de las tablas a mapear de la base de datos.

```
import os
import sys
from sqlalchemy import Column, ForeignKey, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship
from sqlalchemy import create_engine

Base = declarative_base()

class Person(Base):
    __tablename__ = 'person'
    # Here we define columns for the table person
    # Notice that each column is also a normal Python instance attribute.
    id = Column(Integer, primary_key=True)
    name = Column(String(250), nullable=False)

class Address(Base):
    __tablename__ = 'address'
    # Here we define columns for the table address.
    # Notice that each column is also a normal Python instance attribute.
    id = Column(Integer, primary_key=True)
    street_name = Column(String(250))
    street_number = Column(String(250))
    post_code = Column(String(250), nullable=False)
    person_id = Column(Integer, ForeignKey('person.id'))
    person = relationship(Person)

# Create an engine that stores data in the local directory's
# sqlalchemy_example.db file.
engine = create_engine('sqlite:///sqlalchemy_example.db')

# Create all tables in the engine. This is equivalent to "Create Table"
# statements in raw SQL.
Base.metadata.create_all(engine)
```

Ilustración 9 - Modelo ORM de las tablas Person y Address

Fuente*: (Introductory Tutorial of Python's SQLAlchemy, 2013)*

En la Ilustración 10 se puede ver la forma en la que SQLAlchemy permite solicitar información a la base de datos usando las funciones de Python

```
>>> from sqlalchemy_declarative import Person, Base, Address
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///sqlalchemy_example.db')
>>> Base.metadata.bind = engine
>>> from sqlalchemy.orm import sessionmaker
>>> DBSession = sessionmaker()
>>> DBSession.bind = engine
>>> session = DBSession()
>>> # Make a query to find all Persons in the database
>>> session.query(Person).all()
[<sqlalchemy_declarative.Person object at 0x2ee3a10>]
>>>
>>> # Return the first Person from all Persons in the database
>>> person = session.query(Person).first()
>>> person.name
u'new person'
>>>
>>> # Find all Address whose person field is pointing to the person object
>>> session.query(Address).filter(Address.person == person).all()
[<sqlalchemy_declarative.Address object at 0x2ee3cd0>]
>>>
>>> # Retrieve one Address whose person field is point to the person object
>>> session.query(Address).filter(Address.person == person).one()
<sqlalchemy_declarative.Address object at 0x2ee3cd0>
>>> address = session.query(Address).filter(Address.person == person).one()
>>> address.post_code
u'00000'
```

Ilustración 10 - Solicitud de información a la base de datos usando SQLAlchemy

Fuente*: (Introductory Tutorial of Python's SQLAlchemy, 2013)*

- PDF reportlab: es una librería para Python que ayuda en la generación de documentos PDF y gráficos, es muy extensa y tiene muchas funcionalidades desde pequeños textos y figuras hasta ilustraciones y gráficos de gran tamaño.

  Su instalación se hace mediante el administrador de paquetes de Python *"pip package"* utilizando el comando *"pip install reportlab"*, esta librería también cuenta con una API sencilla que permite la creación de documentos PDF desde Python. *(Python, 2018)*

  En la Ilustración 11 se puede ver un ejemplo sencillo del código usado para crear un documento PDF en este caso, vacío.

```
1.  from reportlab.pdfgen import canvas
2.
3.  c = canvas.Canvas("hola-mundo.pdf")
4.  c.save()
```

Ilustración 11 - Creación de un PDF con PDF reportlab

Fuente*: (Python, 2018)*

- PostgreSQL: es un motor de base de datos relacional de código abierto.

Una base de datos relacional es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Las bases de datos de esta naturaleza se basan en el modelo entidad relación, una forma intuitiva y directa de representar datos en tablas. En este tipo de bases de datos cada fila de la tabla es un registro con un ID único llamado *clave*. Las columnas de la tabla contienen atributos de los datos, y cada registro generalmente tiene un valor para cada atributo, lo que facilita el establecimiento de las relaciones entre los puntos de datos. *(¿Qué es una base de datos relacional?, s/f)*

El Diagrama Entidad Relación (ERD), es un diagrama de flujo que puede representar personas, objetos o conceptos que poseen relaciones entre sí dentro un sistema, se utiliza para la creación, depuración y estudio de las bases de datos, cuenta con una simbología en la que cada uno de los elementos representan una característica única.

En la Ilustración 12 se puede ver un ejemplo del modelo entidad relación, en ella se pueden ver algunas entidades como:

- Teacher
- Student
- Groups
- Subjects

También se pueden ver las relaciones que tienen entre ellos.

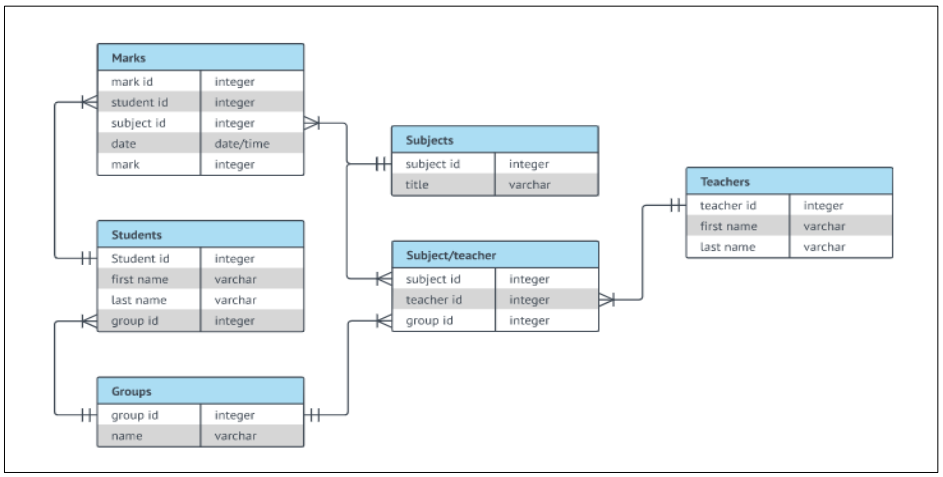


Ilustración 12 - Ejemplo ERD

Fuente: *(Qué es un diagrama entidad-relación, s. f.)*

En la Ilustración 13 se puede ver el significado de la simbología de cada uno de los elementos de relación que aparecen en la Ilustración 12.

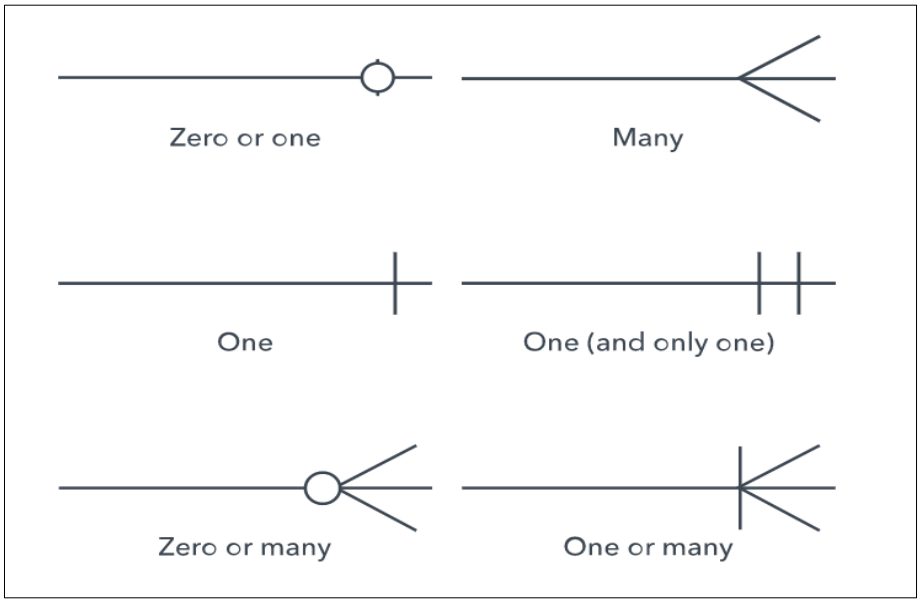


Ilustración 13 - Simbología de ERD

Fuente*: (Qué es un diagrama entidad-relación, s. f.)*

PostgresSQL es un gestor de bases de datos de forma relacional, este gestor es gratuito y de código abierto y tiene más de 30 años de desarrollo que lo ha convertido en una de las mejores opciones del mercado por su fiabilidad, reputación, características y rendimiento. *(¿Nuevo en PostgreSQL?, s/f)*

En la Ilustración 14 se puede ver el Diagrama Entidad Relación de la base de datos de InvIT.

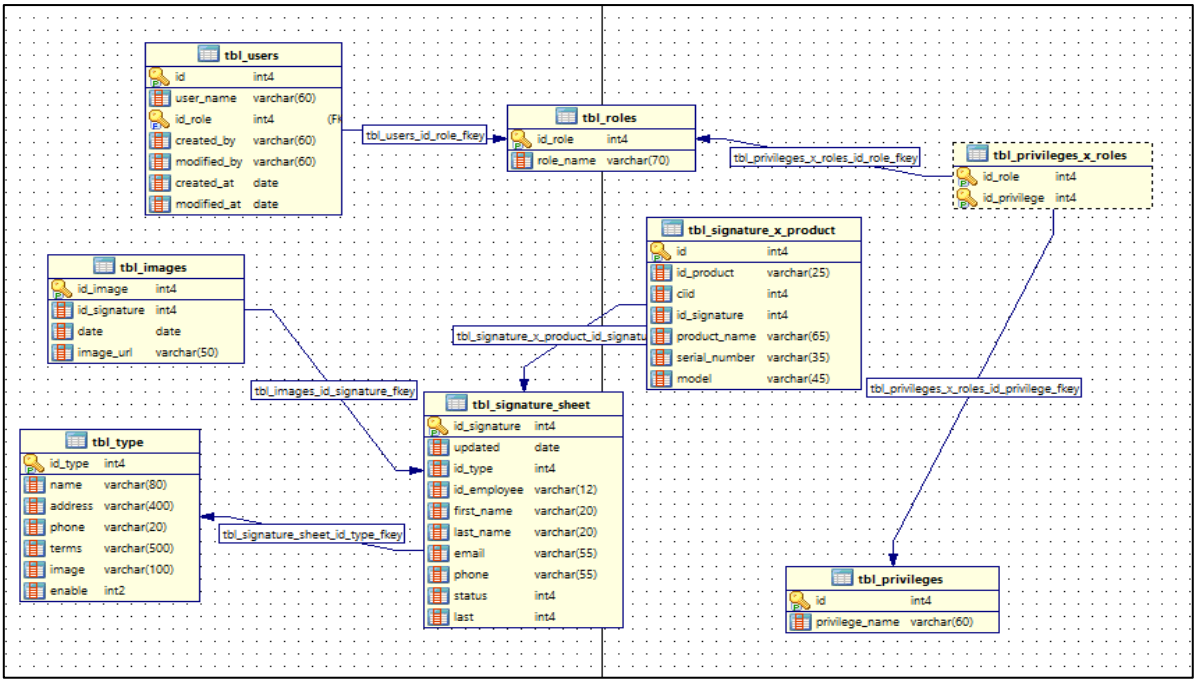


Ilustración 14 - Modelo Relacional de InvIT DB

Fuente: *(Elaboración propia)*

La base de datos para esta versión cuenta con ocho tablas como se muestra en la Ilustración 14, no cuenta con procedimientos almacenados, vistas o Jobs.

Se puede ver una descripción completa de las tablas de esta base de datos en el Anexo 1 Descripción de las tablas de la base de datos de InvIT.

InvIT por tener una base en Angular trabaja con el Modelo Vista Controlador (MVC), el cual representa una ventaja ya que este soluciona uno de los problemas más comunes en el desarrollo de interfaces de aplicaciones porque define las capas de la arquitectura Cliente-Servidor muy famosas en la actualidad. Este patrón separa claramente la parte gráfica de los aplicativos de su parte lógica y de procesos. *(López S, 2009)*

López, S. (2009) menciona que el patrón Modelo Vista Controlador consta de las siguientes partes:

- La vista: es la interfaz gráfica o donde los usuarios pueden ver los resultados de los procesos o peticiones.

- El modelo: este representa a las variables o datos en general que se estén modificando en el momento que se está ejecutando un proceso.
- El controlador: es el encargado de modificar el modelo, en otras palabras, es el que modifica las variables, objetos y los datos en general de acuerdo con lo solicitado por el usuario.

  En la Ilustración 15 se puede ver el patrón Modelo Vista Controlador.

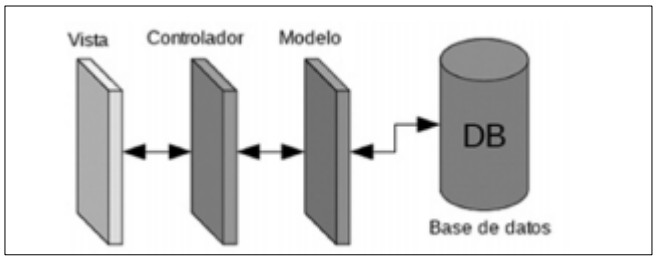


Ilustración 15 - Patrón Modelo Vista Controlador (MVC)

Fuente*: (López S, 2009)*

En el proyecto de InvIT se puede encontrar los archivos que representan el MVC algunos de estos están escritos en TypeScript el cual es una variación del famoso lenguaje de programación llamado JavaScript, pero con características para el desarrollo empresarial. Una de estas características es el tipado fuerte, es decir que a sus variables y objetos se les puede asignar un tipo de dato específico el cual será verificado cada vez que se use, esto no es posible en JavaScript porque está escrito libremente. Otra de las características de TypeScript es que este se compila antes de ejecutarse lo que ayuda a los desarrolladores a la detección de errores antes de su ejecución. TypeScript no se ejecuta en un lenguaje binario, sino que en JavaScript estándar es por eso por lo que se puede ejecutar código TypeScript donde se puede ejecutar JavaScript.

Se utiliza TypeScript cuando se desea utilizar la forma de programar de un lenguaje como Java pero que su código sea ejecutado en JavaScript. *(Fruhlinger, 2020)*

En la Ilustración 16 se puede ver un componente creado en TypeScript y el mismo componente creado en JavaScript, se puede ver también que en el código de TypeScript las variables están tipadas.

greeter.ts

```ts
class Student {
  fullName: string;
  constructor(public firstName, public middleInitial,
  public lastName) {
    this.fullName = firstName + " " + middleInitial +
    " " + lastName;
  }
}

interface Person {
  firstName: string;
  lastName: string;
}


function greeter(person: Person) {
  return "Hello, " + person.firstName + " " +
  person.lastName;
}

var user = new Student("Jane", "B.", "Jones");

document.body.innerHTML = greeter(user);

```

greeter.js

```js
var Student = /** @class */ (function () {
    function Student(firstName, middleInitial, lastName) {
        this.firstName = firstName;
        this.middleInitial = middleInitial;
        this.lastName = lastName;
        this.fullName = firstName + " " + middleInitial + " " +
        lastName;
    }
    return Student;
}());
function greeter(person) {
    return "Hello, " + person.firstName + " " + person.lastName;
}
var user = new Student("Jane", "B.", "Jones");
document.body.innerHTML = greeter(user);

```

Ilustración 16 - TypeScript vs JavaScript

Fuente*: (Intro to TypeScript - DEV, s. f.)*

IV. DESARROLLO

4.1. DESCRIPCIÓN DEL TRABAJO REALIZADO

4.1.1. OBTENER ACCESO E INTRODUCCIÓN A INVIT

Laureate International Universities tiene una estricta política sobre los accesos a sus recursos de software pues no de sus rubros es dedicarse al desarrollo de software y aplicaciones.

El primer paso era solicitar los accesos correspondientes y conocer las políticas de seguridad de Laureate, se me fue otorgado permiso a los repositorios de GitHub bajo una cuenta corporativa con el correo dennis.carcamo@laureate.net, esta introducción a las políticas de seguridad fue impartida por el equipo de IT Operations conformado por el Ing. Kewyn Antonio Medina (jefe Inmediato), Ing. Carlos Ramírez, Ing. Julio Zúniga y el Ing. Oscar Nágera. La reunión se hizo de forma virtual por la plataforma de Microsoft Teams.

Luego de conocer las políticas de seguridad del departamento de IT Operations y de la discusión de los objetivos específicos de la práctica profesional, se me hizo la explicó la plataforma web llamada InvIT, esta es la plataforma a la que se estarán mejorando algunas de sus funcionalidades y agregando otros módulos para enriquecer su funcionalidad.
InvIT tiene un sistema de autenticación tomando las credenciales del Active Directory de la empresa del dominio conocido como "loe.corp", para aquellos usuarios autorizados a poder entrar a InvIT, la autenticación solo es posible si se está haciendo la petición de autenticación dentro de la red de Laureate. En la Ilustración 17 se puede ver la interfaz que Cisco Any Connect usa para la configuración del túnel VPN.

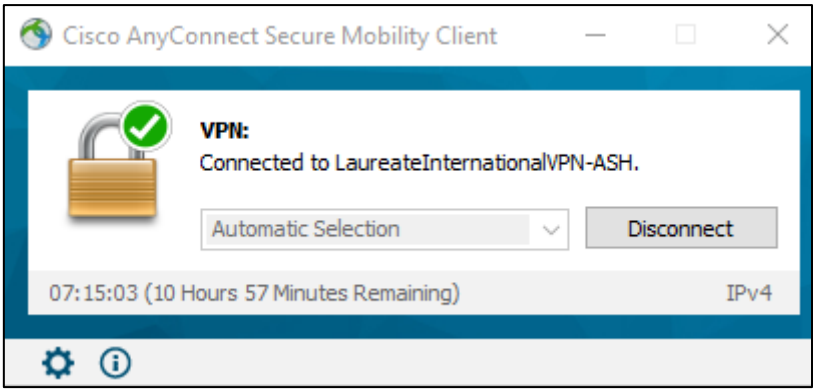


Ilustración 17 - Pantalla de conexión de Laureate VPN

Fuente*: (Cisco AnyConnect Secure Mobility Client, 2020)*

La introducción al Sistema de inventario InvIT fue Impartida por el Ing. Julio Zúniga quien es el autor de la mayor parte de InvIT.

En la Ilustración 18 se puede ver la pantalla de inicio de InvIT, solo el personal autorizado y exclusivamente del departamento de IT tiene acceso.

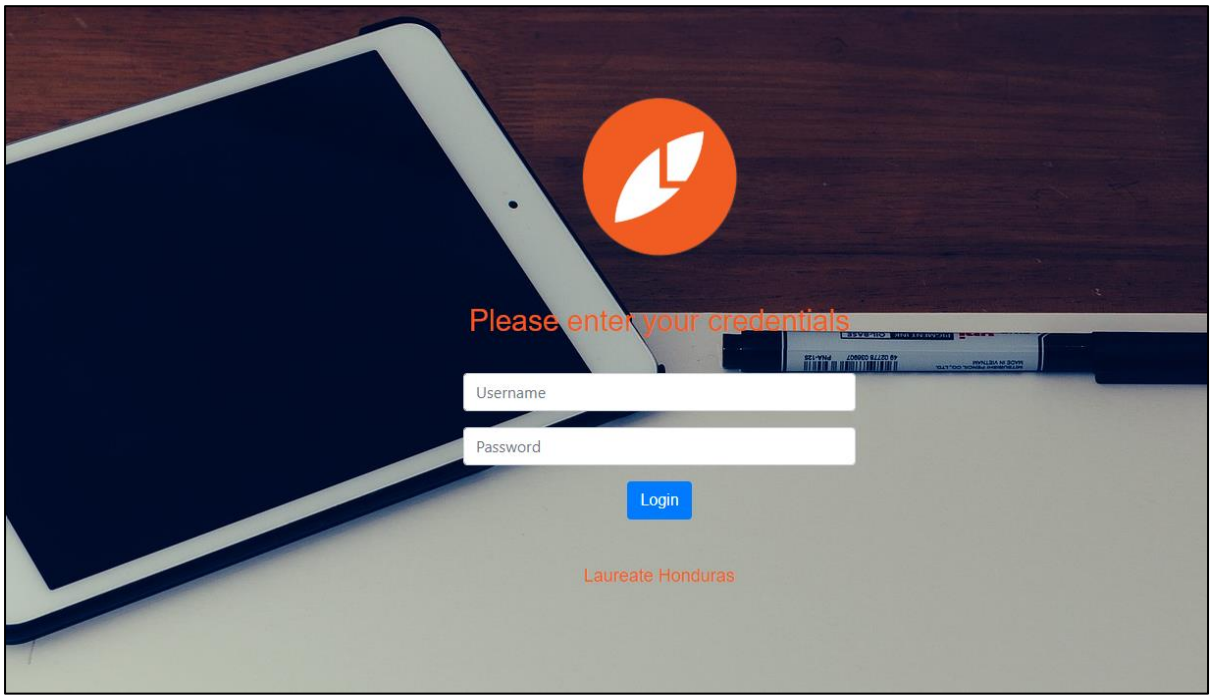


Ilustración 18 - Pantalla para inicio de sesión InvIT

Fuente*: (InvIT, 2020)*

En la Ilustración 19 se puede ver la página de aterrizaje de InvIT, esta es la primera vista que el usuario tiene después de autenticarse.

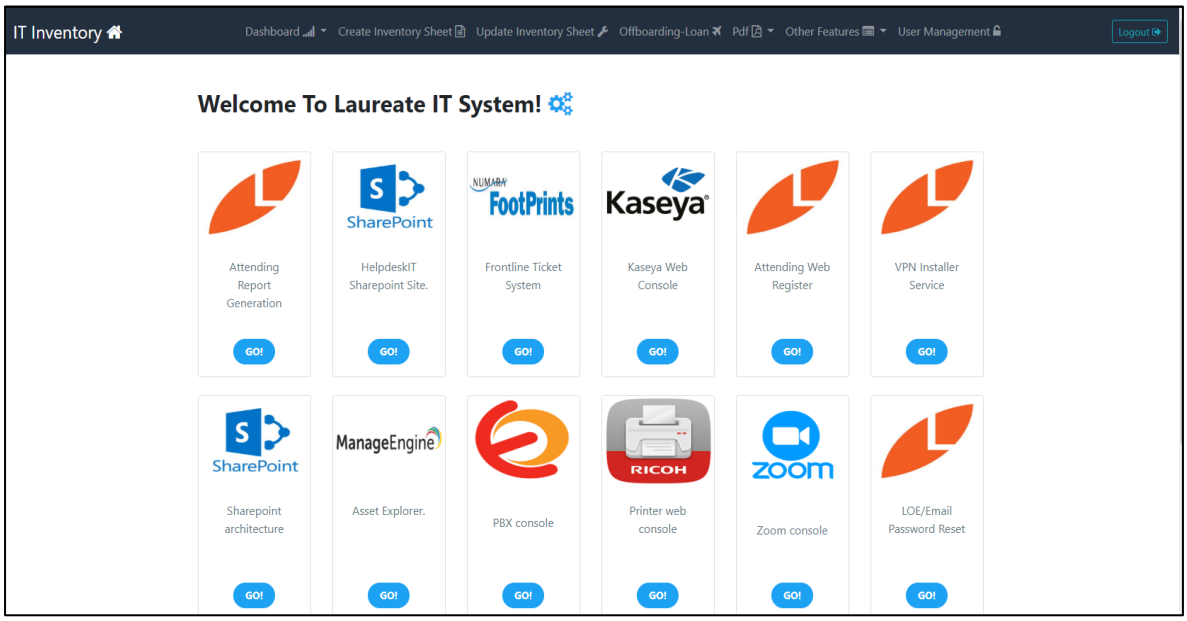


Ilustración 19 - Página principal de InvIT

Fuente*: (InvIT, 2020)*

En la Ilustración 20 se puede ver el resultado de hacer una búsqueda por número de hoja de inventario para realizar actualizaciones.

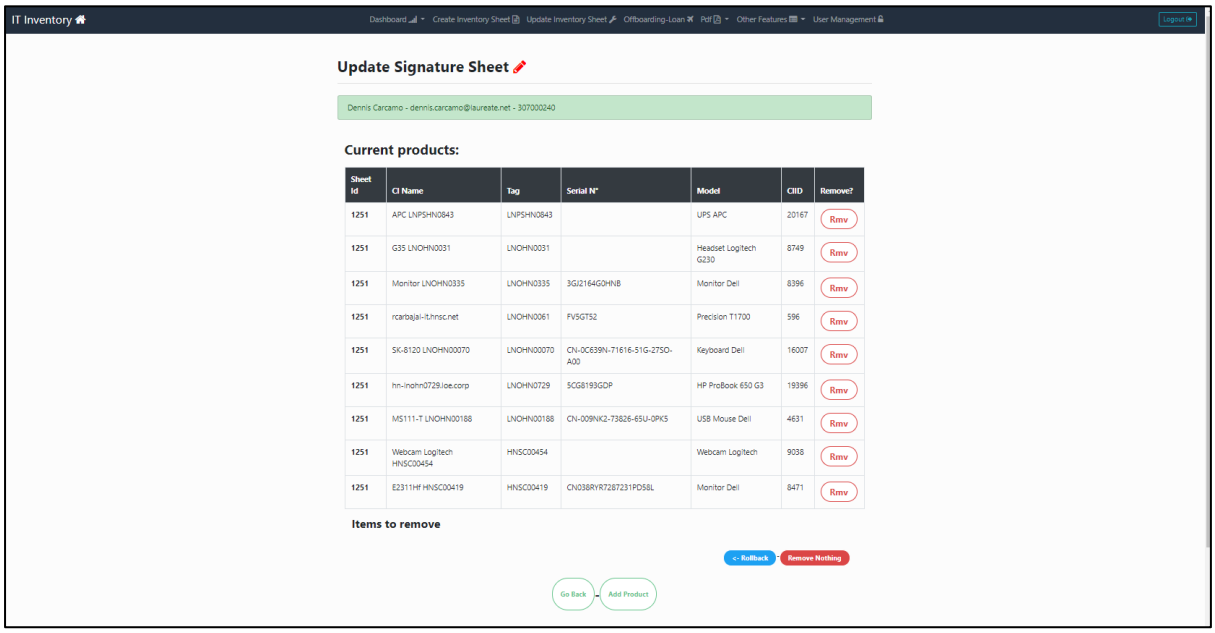


Ilustración 20 - Pantalla del módulo de actualizar hoja de inventario

Fuente*: (InvIT, 2020)*

En la Ilustración 21 se puede ver la vista para la administración de las hojas de inventario.

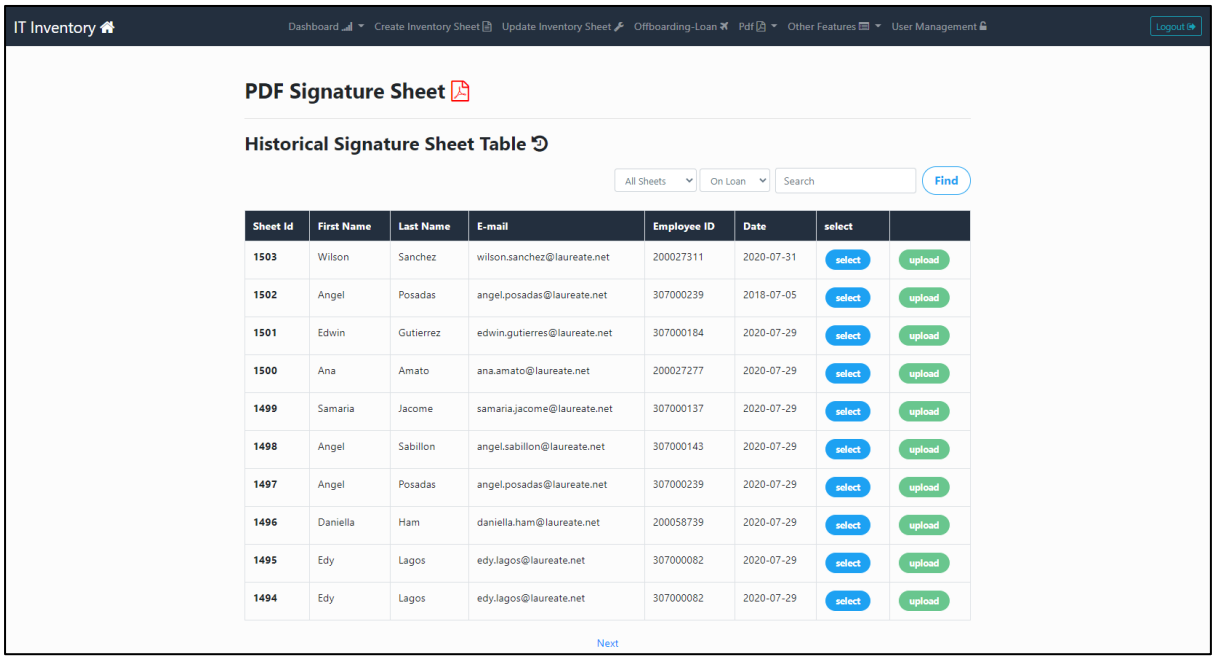


Ilustración 21 - Vista para la administración de las hojas de inventario

Fuente*: (InvIT, 2020)*

Hay que recordar que InvIT es un orquestador entre AssetExplorer y las gestiones de inventario que involucren a los usuarios, como ser asignación de equipo, retiro de equipo, gestiones de préstamo, compra de equipos y software, entre otros.

4.1.2. ACCESO A BACKEND Y FRONTEND DE INVIT

Para poder realizar esta tarea era indispensable el conocimiento de la funcionalidad de GitHub pues los repositorios del código fuente de InvIT están almacenados aquí bajo la cuenta corporativa de Laureate International Universities.

Después de ejecutar el comando "git clone" se podía ver el código fuente de InvIT, código un poco confuso en la forma que estaba pues en una sola carpeta se encontraba el frontend, el backend y la versión de producción de la aplicación.

Se dedicó parte del día para entender cuál era la estructura general de la aplicación y qué cosas debía tomar en consideración antes de hacer las instalaciones de las dependencias necesarias.

Era necesario el conocimiento de Pip Package para Python pues las dependencias del backend se instalaban con este administrador de paquetes. Uno de los puntos muy importantes a la hora de trabajar con Python es la utilización de ambientes virtuales, estos permiten instalar dependencias de Python solamente en el proyecto que se está trabajando, de lo contrario las dependencias se instalan en el sistema operativo y esto puede causar el fallo en otras aplicaciones o problemas para desarrollar otro software que necesite versiones diferentes de la dependencia.

Para crear un entorno virtual para el proyecto de Laureate se usó el comando: "*py -m venv Laureate*" y luego para activar el ambiente virtual se debe correr el comando *"source C:/User/dejoc/Downloads/Laureate/Scripst/actívate*".

Las dependencias del backend debían instalarse una a una hasta que se satisficieran todas las dependencias para que el backend pueda ejecutarse.

En la Tabla 1 se listan las dependencias que se instalaron para el correcto funcionamiento del backend.

Tabla 1 - Dependencias del Backend

Dependencia	Versión
alembic	0.9.6
aniso8601	1.3.0
astroid	1.6.0
certifi	2017.11.5
chardet	3.0.4
click	6.7
colorama	0.3.9
Flask	0.12.2
Flask-JWT	0.3.2
Flask-JWT-Extended	3.8.1

Dependencia	Versión
flask-marshmallow	0.8.0
Flask-Migrate	2.1.1
Flask-RESTful	0.3.6
Flask-Script	2.0.6
Flask-SQLAlchemy	2.3.2
Flask-WTF	0.14.2
idna	2.6
isort	4.2.15
itsdangerous	0.24
Jinja2	2.10
lazy-object-proxy	1.3.1
ldap3	2.4.1
Mako	1.0.7
MarkupSafe	1.0
marshmallow	2.15.0
marshmallow-sqlalchemy	0.13.2
mccabe	0.6.1
pdfkit	0.6.1
Pillow	5.0.0
psycopg2	2.7.3.2
pyasn1	0.4.2
pyasn1-modules	0.2.1
PyJWT	1.4.2
pylint	1.8.1
python-dateutil	2.6.1
python-editor	1.0.3
pytz	2017.3
reportlab	3.4.0
requests	2.18.4
six	1.11.0

Dependencia	Versión
SQLAlchemy	1.2.2
urllib3	1.22
virtualenv	15.1.0
virtualenvwrapper-win	1.2.5
webargs	1.8.1
Werkzeug	0.14.1
wrapt	1.10.11
WTForms	2.1

Fuente: *(Elaboración propia)*

Para las dependencias del front-end solamente es necesario correr el comando *"npm install"* porque en Angular este comando instala todas las dependencias automáticamente.

Para probar la funcionalidad del frontend se debe ejecutar el comando "ng serve" y se verá que se ejecuta en el localhost, en este caso usando el puerto 4200.

Para probar la funcionalidad del backend se debe correr el comando "Python App.py" donde App.py es el nombre del archivo. Este generalmente corre en el localhost.

En la Ilustración 22 se muestra en un árbol la estructura principal del backend de InvIT para tener una idea más clara en qué parte se utilizan las dependencias instaladas.

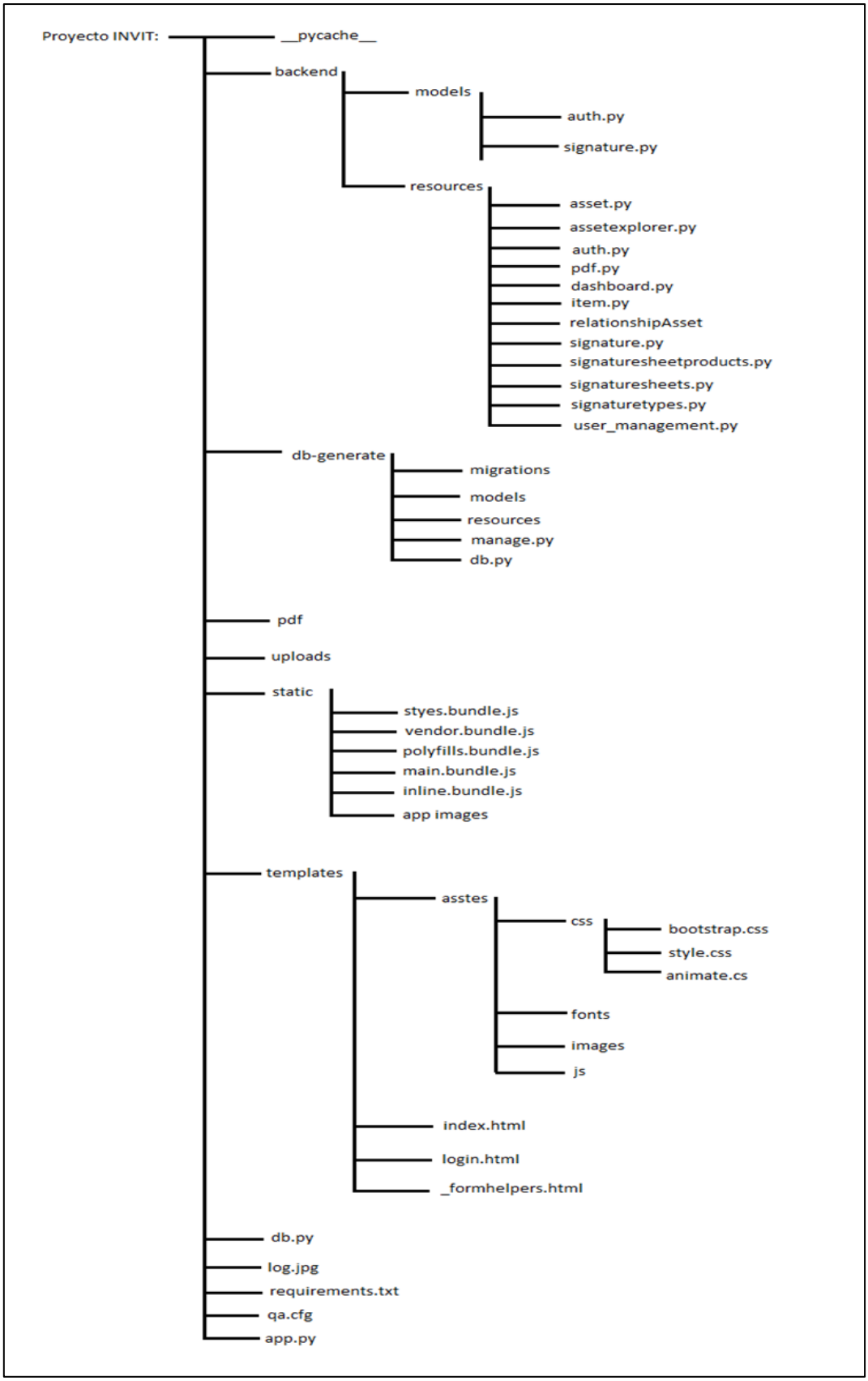


Ilustración 22 - Estructura principal del Backend de InvIT

Fuente*: (Arquitectura de InvIT, 2019)*

Una vez que se tenían todas las dependencias del frontend y del backend instaladas se debía probar la funcionalidad de ambos para que se pueda empezar a modificar y agregar código.

4.1.3. RESTAURACIÓN DE BASE DE DATOS DE INVIT

Las bases de datos juegan un papel esencial en el sistema de inventario InvIT, es la fuente que alimenta todo el sistema y la información que se desea visualizar y consultar de forma amigable y fácil.

Para que InvIT funcione se necesitan dos bases de datos, la base de datos propia del sistema de InvIT y la base de datos que maneja el AssetExplorer de la compañía, ambas bases de datos están en PostgreSQL alojadas en un servidor de producción de Laureate y una copia para pruebas en un servidor de QA.

Para poder empezar el desarrollo de los módulos es necesario tener acceso a las bases de datos, para así poder crear los mapeadores de objetos relacionales (ORM) con SQLAlchemy y aprovechar toda la funcionalidad del lenguaje SQL.

La extracción y restauración de la base de datos de InvIT fue de las más sencillas pues su base de datos es pequeña y tiene menos de 10,000 registros por ser un software relativamente nuevo, para poder tener una instancia de esta base de datos se siguieron los siguientes pasos.

- Crear una copia de seguridad de las bases de datos InvIT que se encuentra en producción. En la Ilustración 23 se puede ver la opción elegida para realizar la tarea.

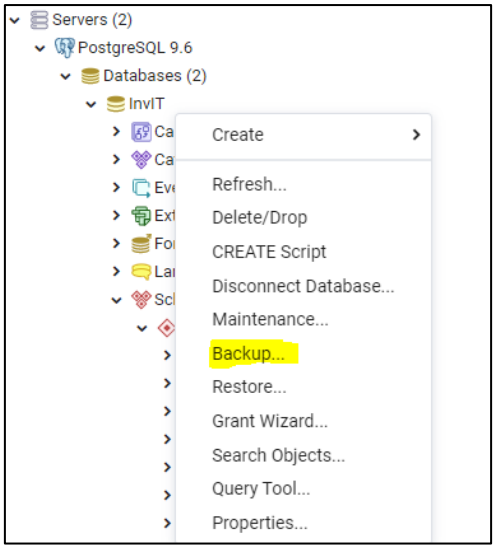


Ilustración 23 - Backup de InvIT DB

Fuente*: (PgAdmin, 2020)*

- Restauración de la base de datos de InvIT en un servidor PostgreSQL local.

  Se debe tener el archivo *"invit-database-202007211736.backup"* este archivo es creado por PG Admin al momento de realizar un respaldo de la base de datos, el archivo fue compartido por el administrador de las bases de datos para que se pueda implementar en el servidor de PostgreSQL local.

  En la Ilustración 24 se puede ver la opción utilizada para hacer la restauración.

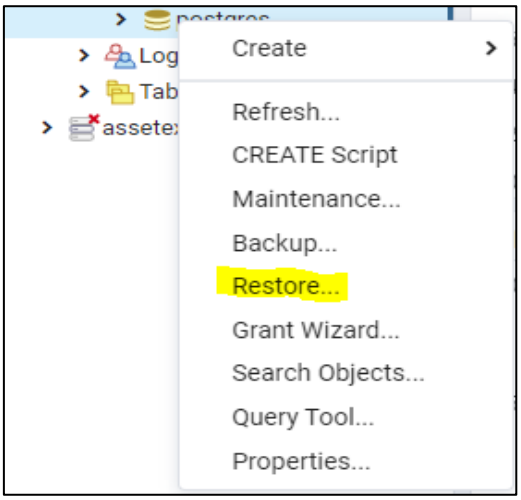


Ilustración 24 - Restauración de InvIT DB en servidor local

Fuente*: (PgAdmin, 2020)*

4.1.4. CONEXIÓN DE BASE DE DATOS REMOTA A ASSETEXPLORER DB

La base de datos de Asset Management era más difícil de poder implementarla en forma local pues esta base de datos es mucho más antigua y está bastante poblada, tanto era así que PGAdmin no podía completar la extracción de la copia de restauración.

La solución temporal fue crear una conexión directa desde PGAdmin a la base de datos de Asset alojada en el servidor de QA y así poder consumir sus datos, en la Ilustración 25 se puede ver la ventana de configuración para la conexión al servidor de PostgreSQL.

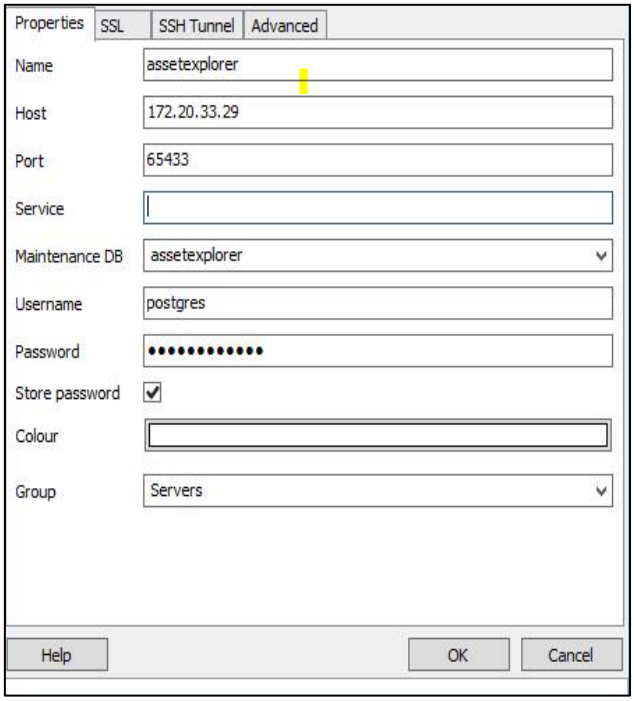


Ilustración 25 - Configuración de conexión a Asset Management DB

Fuente: (PgAdmin, 2020)

Una vez colocada la información de conexión se puede acceder a la base de datos de AssetExplorer, es importante mencionar que solo se puede conectar a la base de datos se está utilizando el túnel VPN de Laureate. En la Ilustración 26 se puede ver la base de datos de AssetExplorer agregada.

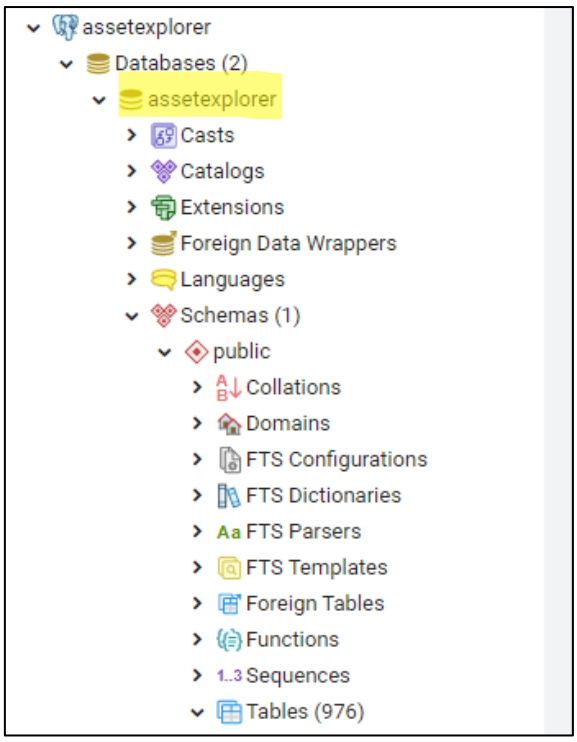


Ilustración 26 - AssetExplorer DB

Fuente: *(PgAdmin, 2020)*

El problema con esta base de datos se debe a que los servidores que permiten conexiones externas tienen brechas de seguridad que el equipo de seguridad ha reportado y prohíben las conexiones externas a los servicios dentro del servidor, el administrador de la base de datos tuvo que hacer una excepción y colocar el punto de acceso en la lista blanca para que se permitiera la conexión.

4.1.5. Estudio de los modelos ORM de InvIT

SQLAlchemy cambia la forma tradicional en la que se trabaja con las bases de datos, este conjunto de herramientas es famoso por la manera en que mapea en forma de objetos relacionales (ORM).

Era fundamental conocer los modelos ORM para conocer qué tipo de consultas se están haciendo en InvIT actualmente. Esto con el fin de entender la forma en que los modelos ORM hacen consultas a la base de datos.

Como se ha mencionado la base de Datos de InvIT consta de pocas tablas y son estas la que utilizan el modelo ORM, en la Ilustración 27 se puede ver un fragmento del código en que muestra el modelo ORM de una de las tablas de la base de datos.

```python
class PrivilegesModel(db.Model):
    __tablename__ = 'tbl_privileges'
    id =    db.Column(db.Integer, autoincrement = True,primary_key = True)
    privilege_name = db.Column(db.String(60), nullable = False)

    def __init__(self, name):
        self.privilege_name = name

    def json(self):
        return{
                'id': self.id,
                'privilege_name': self.privilege_name
            }

    def insert(self):
        db.session.add(self)
        db.session.commit()


    @classmethod
    def bring_all(cls):
        return cls.query.all()

    def delete_item(self):
        db.session.delete(self)
        db.session.commit()
```

Ilustración 27 - Modelo ORM de la tabla privilegios de InvIT DB

Fuente: *(Elaboración propia)*

Se puede observar que existe una función llamada "*insert*" encargada de traducir del modelo ORM a lenguaje SQL la sentencia "*insert into*". De esta forma SQLAlchemy facilita a los programadores a centrarse exclusivamente en el desarrollo y no en la funcionalidad de la base de datos. En la Ilustración 28 se puede ver uno de los ORM de la base de datos de InvIT.

```python
from db import db
from sqlalchemy import Table, Column, Float, Integer, String, MetaData, ForeignKey
from app import app

class ItemModel(db.Model):
    __tablename__ = 'tbl_items'
    name = db.Column(db.String(60), primary_key = True)
    price = db.Column(db.Float(2), unique = False)

    def __init__(self, name,price):
        self.name = name
        self.price = price

    def insert(self):
        db.session.add(self)
        db.session.commit()


    @classmethod
    def find_by_name(cls, name):
        return cls.query.filter_by(name=name).first()

    def json(self):
        return {'name':self.name, 'price':self.price}

    @classmethod
    def bring_all(cls):
        return cls.query.all()

    def delete_item(self):
        db.session.delete(self)
        db.session.commit()
```

Ilustración 28 - ORM de la tabla ItemModel de InvIT DB

Fuente*: (Elaboración propia)*

4.1.6. ESTUDIO DEL CÓDIGO ACTUAL DE INVIT

4.1.6.1. Archivo de configuración

La aplicación cuenta con un archivo de configuración, en este el administrador establece los siguientes parámetros:

- URLDB: base de datos AssetExplorer, usuario y contraseña
- ASSETDB: base de datos local InvIT, usuario y contraseña
- UPLOAD_FOLDER: ruta donde guardar archivos .pdf cargados desde el sistema.
- PDFPATH: ruta donde guardar los .pdf generados por el sistema.
- PDFIMAGEPATH: ruta de la imagen del logo de Laureate.

- FRONTENDIMAGESPATH: ruta de las imágenes para cargar al navegador.

4.1.6.2. Cargar Index y JS

Quien regresa el index.html al browser es el app.py este lo busca en la carpeta llamada "*templates"*, pero es necesario que el documento index.html tenga una sintaxis para que pueda cargar sus archivos, esta sintaxis se aplica dentro de las etiquetas script al final del documento como se muestra en el siguiente código.

```
<script
    type="text/javascript"
    src="{{url_for('static',filename='inline.bundle.js')}}">
</script>
```

Esta sintaxis le indica al índex que proceda a buscar sus archivos en la carpeta llamada "*static*". De lo contrario se obtendrá un 404 de archivos no encontrados.

4.1.6.3. Carpetas y Documentos

Es importante mencionar algunos de las carpetas y archivos de InvIT para mantener su correcta documentación y entender su funcionalidad, algunos de estos son:

- db-generate: los documentos en esta carpeta ayudan a la creación de la base de datos, utilizando el documento manage.py, este procedimiento se detallará más en la sección de implementación.
- Backend: contiene dos carpetas más, la primera que se llama "*models*", donde se presentan los modelos de la base de datos, y la segunda que se llama "*resources*", donde se encuentran los recursos con los que trabajarán todos los "*endpoints*" del sistema.
- Pdf: contiene todos los archivos .pdf creados por el sistema.
- Uploads: contiene los .pdf escaneados por los usuarios al sistema.
- Static: es la carpeta con el nombre de "*static*", esta tiene todos los archivos javascript para el funcionamiento del frontend.
- Templates: contiene los archivos de Lenguaje de Marcas de Hipertexto (HTML) para cargar la página en el navegador, de igual manera los diferentes recursos y archivos que los HTML necesitan, como ser los .css, .js, imágenes y animaciones.

- db.py: documento en Python que facilita la conexión con la base de datos.
- Requirements.txt: lista de las diferentes dependencias que necesita el sistema para su buen funcionamiento.
- qa.cfg: es un archivo de configuración utilizado por el menú o página principal de la aplicación, para cargar información establecida por el administrador de servidor.
- app.py: es el documento principal del sistema con lo que trabaja y en lo que se integra todo lo demás.

4.1.7. CREACIÓN ARCHIVOS DE CONFIGURACIÓN PARA LOS DIFERENTES AMBIENTES

Al momento que se desarrolla una aplicación o herramienta de software es necesario tener en consideración el uso de varios ambientes en los que se pueda poner a prueba el funcionamiento del aplicativo, esto se hace con el fin de poder probar todos los posibles casos de uso que se le puede dar al software en desarrollo y que este siempre cumpla con su trabajo.

En la actualidad las nuevas prácticas de programación sugieren que se tenga a disposición diferentes ambientes de desarrollo que ayuden a asegurar el funcionamiento total del software y que se pueda sacar al mercado un producto con altos estándares de calidad.

Generalmente son tres los ambientes en los que el software se puede encontrar: el ambiente de desarrollo, el ambiente de QA y el ambiente de producción; este último es cuando el software ya está puesto en funcionamiento y realizando el trabajo para el que fue pensado.

InvIT no es la excepción, así como el software se puede ejecutar en los diferentes ambientes, este software también posee sus respectivos ambientes, el problema era que la misma versión del software para el ambiente de producción era la misma que se utiliza para el ambiente de desarrollo y de QA en lo único que difieren es en la base de datos.

El problema antes mencionado se traducía a un problema con las versiones del software y con modificaciones en el código recurrentes que se podrían evitar, todos los archivos del frontend eran modificados manualmente cada vez que se quería hacer un lanzamiento en uno de los ambientes.

Fue sugerido el uso de archivos de configuración para que al hacer un pequeño cambio en una variable de configuración el proyecto completo sepa para qué ambiente específico se está trabajando.

Para que la tarea se pudiera completar era necesario fijarse en la estructura del frontend porque es en esta parte donde los complementos y vistas se restringen al usuario de acuerdo con su nivel de permiso. En la Ilustración 29 se puede ver la forma en la que cada componente comprobaba si debía mostrarse al usuario o no; esto último era gestionado por la variable llamada *"privilege"*, también se puede ver que si se deseaba hacer una versión de QA o de producción estos permisos se tenían que comentar o eliminar.

```
let val = this.cookieService.get("token");
if (val) {
  let djtw = jwt_decode(val);
  let identity = djtw["identity"];
  for (var _i = 0; _i < identity.length; _i++) {
    if (identity[_i]["privilege"] == "dashboard") {
      this.privilege = true;
    }
  }

  if (this.privilege) {
    this.initialDashboard();
    this.getCrosscheck();
  } else {
    window.location.href = "/login";
  }
} else {
  //comentar los privilegios en producción y qa;
  this.privilege = true;
  this.initialDashboard();
  this.getCrosscheck();

  //OJO descomentar el href;
  window.location.href = "/login";
}
```

Ilustración 29 - Evaluación de privilegios antes del cambio

Fuente: *(Elaboración propia)*.

Se utilizaba un servicio de cookies de la librería "*ngx-cookie-service*" para que alojara las credenciales del usuario que estaba dentro de la sesión, el problema radica en que la base de datos que se usa en ambiente de desarrollo no guardaba credenciales de los usuarios autorizados del sistema por lo que la verificación de las credenciales no se podía hacer y por no poder, los componentes se quedaban bloqueados haciendo imposible la visualización de los elementos renderizados al momento de la ejecución.

Lo primero que se hizo fue crear un nuevo archivo de configuración llamado *"appEnviromentConfig"* que tenía una variable llamada "*enviroment*" a la que se le debía asignar el valor *"dev"* si se estaba trabajando la aplicación o asignarle *"prod"* si se estaba trabajando para la versión de lanzamiento. En la Ilustración 30 se puede ver el contenido de la clase de configuración.

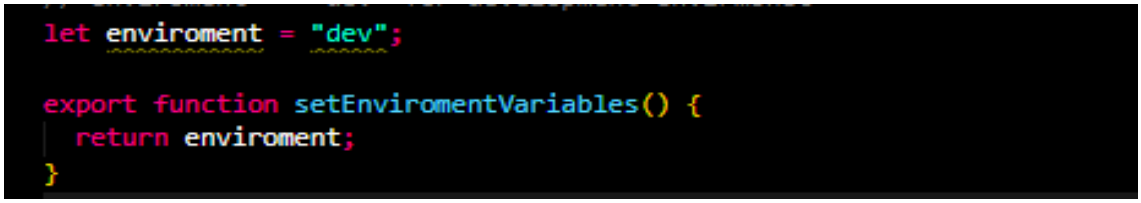


Ilustración 30 - Clase de Configuración que guarda la variable "enviroment"

Fuente: *(Elaboración propia)*

Como segundo paso se modificó en cada uno de los archivos del frontend la evaluación de los privilegios con la sentencia de decisión "if" y así poder decirles a todos los archivos en qué ambiente están y cómo deben comportarse. En la Ilustración 31 se puede ver que ahora lo primero que hace cada componente es llamar a la función *"setEnviromentVariables()"* que devuelve como respuesta el valor que tiene la variable *"enviroment"*, si tenía un valor verdadero el componente debía mostrarse, pero si su valor era falso no debía mostrarse.

```
ngOnInit() {
  if (setEnviromentVariables() == "dev") {
    this.privilege = true;
    this.initialDashboard();
    this.getCrosscheck();
    //window.location.href = "/login";
  } else if (setEnviromentVariables() == "prod") {
    let val = this.cookieService.get("token");
    if (val) {
      let djtw = jwt_decode(val);
      let identity = djtw["identity"];
      for (var _i = 0; _i < identity.length; _i++) {
        if (identity[_i]["privilege"] == "dashboard") {
          this.privilege = true;
        }
      }

      if (this.privilege) {
        this.initialDashboard();
        this.getCrosscheck();
      } else {
        window.location.href = "/login";
      }
    }
  }
```

Ilustración 31 - Evaluación de privilegios después del cambio

Fuente: *(Elaboración propia)*

Esta modificación a los archivos del frontend ayudará a los futuros desarrolladores de InvIT a poder cambiar de ambiente el aplicativo de forma fácil y rápida.

4.1.8. CREACIÓN DE INTERFAZ GRÁFICA DE LA REPORTERÍA PERSONALIZABLE

Uno de los objetivos de la práctica profesional era la elaboración de un módulo de reportería personalizable que permitiera al usuario la suficiente flexibilidad para solicitar reportes de la manera deseada. Era necesario saber que los reportes de los que deseaban siempre estaban referidos a la base de datos de los activos de la empresa y los colaboradores.

4.1.8.1. Creación de los formularios HTML.

Lo primero que se debía hacer era la estructura HTML a la que se le daría el estilo luego con Bootstrap. El componente está pensado para realizar reportería personalizable sobre:

- Colaboradores: en este complemento se quería poder obtener información relevante de todos los colaboradores de Laureate y existía la posibilidad de agregar campos que muestren la relación de ellos con los activos.
- Hojas de inventario: en este complemento se espera poder hacer solicitudes de información que involucren las páginas de inventario en la que se lleva un registro de todos los elementos asignados y en préstamo.
- Activos: en este componente de espera hacer reportes de todo el activo en general, desde los que están en la bodega hasta los que están en préstamo o asignados a un usuario.

Una vez creada la plantilla HTML se utilizó la librería de frontend muy conocida llamada Bootstrap que permite aplicar estilos predefinidos al componente de HTML siempre dejando la posibilidad de hacer cambios personales si así se desea. En la Ilustración 32 se puede ver la vista de los reportes de los empleados.

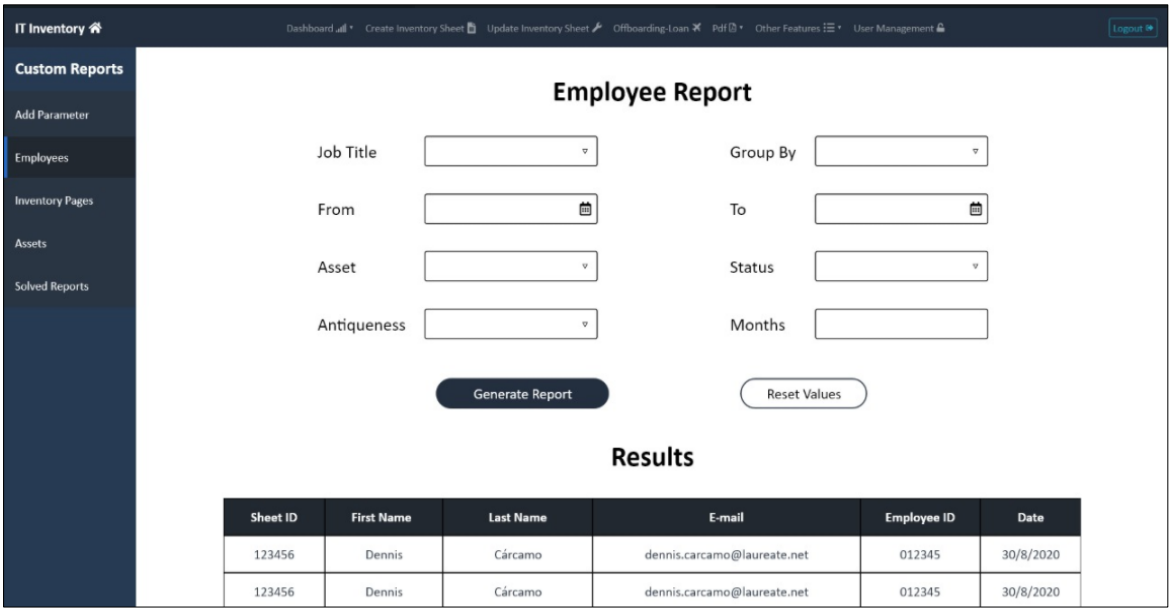


Ilustración 32 - Vista del componente de reportería

Fuente: *(Elaboración propia)*

4.1.8.2. Unión de directivas de Angular con HTML

Después de la creación de la estructura HTML era necesario hacer la unión del archivo .html con el archivo de TypeScript que es el encargado de llevar toda la funcionalidad de los componentes anteriormente mencionados. Se utilizaron las directivas estructurales de Angular tales como:

- NgIf: esta directiva permite el bloqueo o desbloqueo de los componentes.
- Ngfor: esta directiva renderiza elementos iterativos.
- NgValue: esta directiva modifica el valor de la variable dependiendo de un evento.
- NgModel: es bidireccional, aplica los valores a las variables correspondientes.

Estas se usan para poder unir los valores de los formularios del backend con los valores de las variables del frontend.

Uno de los problemas que se encontró fue que los valores para los campos de selección múltiple de los formularios no debían ser colocados en el código HTML, sino que se debía encontrar la forma de hacerlo lo más dinámico posible y para poder hacer eso se utilizó en este punto la directiva ngFor; esta directiva permite renderizar elementos iterativos como lo es el arreglo de campos.

También era necesario crear las funciones de mostrar y bloquear los componentes, esto se lograba con la directiva ngIf unidas a una variable booleana que era administrada por el TypeScript.

En la Ilustración 33 se puede ver una sección del código de uno de los componentes, de un lado el HTML y el otro lado en TypeScript, se puede ver la unión y la referencia a la variable "*jobtitle"* por medio de las directivas estructurales de Angular, en este caso se ve la función de *"[(ngModel)]"*.

En el HTML:

```
<div class="form-inline">
  <select
    class="form-control"
    [(ngModel)]="jobtitle"
    (change)="($event.target.value)"
  >
    <option *ngFor="let job of jobtitleOptions" [ngValue]="job">
      {{ job.job }}
    </option>
  </select>
</div>
```

En el TypeScript :

```
jobtitle: any;
jobtitleGroup: any;
```

Ilustración 33 - Usando las directivas Estructurales de Angular

Fuente: (*Elaboración propia)*

4.1.9. CREAR LAS PETICIONES HTTP Y CONSULTAS A LAS BASES DE DATOS.

Las peticiones HTTP son las que permiten a la aplicación obtener la información desde el backend y de las bases de datos.

En primera instancia se debe crear un servicio con el *comando "ng new service servicename"*, esto se hace por estandarización al momento de escribir código en TypeScript porque facilita la depuración y la detección de errores.

Se creó el primer servicio para la generación de reportes de los empleados, esta función recibe 8 parámetros de los cuales 7 son de tipo texto y uno es de tipo numérico. Esta función va a retornar el resultado que el backend devuelva después que se le solicite la información.

En la Ilustración 34 se puede ver el servicio de reportes y la petición http que hace al backend utilizando la URL *"http://127.0.0.1:5000/api/v1/employeesreports"* y un cuerpo de información que será procesada por el backend.

```
export class CustomReportsService {
  constructor(private httpClient: HttpClient) { }

  employeeReports(
    jobtitle: string,
    jobtitleGroup: string,
    jobtitleFrom: Date,
    jobtitleTo: Date,
    jobtitleAsset: string,
    jobtitleStatus: string,
    jobtitleRange: string,
    jobtitleMonths: number
  ) {

    // tslint:disable-next-line: max-line-length
    return this.httpClient.post(`http://127.0.0.1:5000/api/v1/employeesreports`,{
    jobtitle: jobtitle,
    jobtitleGroup: jobtitleGroup,
    jobtitleFrom: jobtitleFrom,
    jobtitleTo: jobtitleTo,
    jobtitleAsset: jobtitleAsset,
    jobtitleAssetStatus: jobtitleStatus,
    jobtitleAssetRange: jobtitleRange,
    jobtitleAssetMonths: jobtitleMonths})
    .map((result) => result);
  }
```

Ilustración 34 - Servicio para generar reportes de empleados

Fuente: *(Elaboración propia)*

Se empezó a trabajar con el método *"get"* pero este presentó problemas al momento de enviar parámetros por el URL o como *"querystring"*, esto hizo que se utilizara otro método llamado *"post"* el cual permitía el envío de parámetros por medio de un payload o cuerpo en el que se puede enviar una Notación de Objeto de JavaScript (JSON) y de este extraer los valores para poder realizar la solicitud de información a la base de datos y retornar los resultados.

En el backend de la aplicación debería existir una clase a la que el URL del servicio hace referencia, de esta forma la aplicación sabe a qué clase se le están enviando los parámetros

Una vez que el backend era llamado para que procesara la petición; este extraía la información que el cuerpo de la solicitud traía, indicando anticipadamente qué campos vendrán y qué tipo de datos se esperaba en cada campo. En este punto ya se tenía toda la información necesaria para que se creara la consulta necesaria y mandarlo a la base de datos de asset explorer para así obtener la información solicitada.

Para que se viera ordenada la consulta, se fue creando por secciones basados en los parámetros que se recibieron. En el Anexo 2 - Consulta para los reportes de los empleados, se puede ver la consulta completa.

El resultado de la consulta enviaba al frontend la información en formato JSON, el cual se transformaba a un arreglo de objetos para que así se pudiera mostrar en forma de tabla al usuario que solicitó la información.

En la Ilustración 35 se puede ver la transformación del resultado a un arreglo en el frontend.

```
let res = result["values"];
this.jobtitleResult = Object.keys(res).map((e) => res[e]);
this.showEmployeeTable = true;
```

Ilustración 35 - Transformación de JSON a un arreglo

Fuente: *(Elaboración propia)*

Después de transformar la respuesta a un arreglo, se podía mostrar al usuario utilizando las directivas estructurales de Angular, en este caso utilizando *"*ngFor"*. En la Ilustración 36 se puede ver cómo se mostró el arreglo en formato de tabla para la visualización del usuario.

```
<tbody *ngFor="let value of assetResults">
  <tr class="select">
    <th scope="row">{{ value.resourceid }}</th>
    <td>{{ value.assettag }}</td>
    <td>{{ value.serialno }}</td>
    <td>{{ value.componentname }}</td>
    <td>{{ value.resourcename }}</td>
    <td>{{ value.displaystate }}</td>
```

Ilustración 36 - Mostrando el resultado en forma de tabla

Fuente*: (Elaboración propia)*

En la Ilustración 37 se pueden ver los campos de la tabla que se muestran al usuario con el código representado en la Ilustración 36.

Employee ID	First Name	Last Name	Department
307000220	Daniela	Ramirez	Academic Quality & Cont Improvement/ Teacher Develop & Support
307000282	Ilich	Garcia	Advantage Courseware
307000298	Karol	Valdivia	Comunications

Ilustración 37 - Tabla de resultados para los reportes de empleados

Fuente: *(Elaboración propia)*

De la misma forma que se creó el servicio para los reportes de los empleados, se creó el servicio para los reportes de los activos de Laureate. Se puede ver la consulta SQL que se utilizó para solicitar la información a la base de datos de asset explorer en el Anexo 3 - Consulta SQL para los reportes de los activos.

En este servicio también se convirtió la respuesta en un arreglo de objetos para que se pudiera mostrar al usuario en forma de tabla. En la Ilustración 39 se puede ver el código de lo descrito anteriormente.

```
let res = result['values']
this.assetResults = Object.keys(res).map((e) => res[e]);
this.showAssetTable = true;
```

Ilustración 38 - Creación del Arreglo de Resultados para los activos

Fuente: *(Elaboración propia)*

En la Ilustración 39 se puede ver la tabla que se le muestra al usuario con la información solicitada y procesada por el backend y el frontend.

Resource ID	Asset Tag	Serial Number	Component Name	Resource Name
681136	lnohn0831	5CG9238678	Hp ZBook 14u G5	hn-lnohn0831.loe.c
681576	lnohn0829	5CG9238679	Hp ZBook 14u G5	hn-lnohn0829.loe.c
721348	LNOHN1981	C02ZJK0SMD6M	MacBook Pro (16 inch, 2019)	MacBookPro (16 inc
722267	LNOHN1982	LNOHN1982	MacBook Pro (16 inch, 2019)	MacBoock Pro (16 i

Ilustración 39 - Tabla de resultados para los reportes de los activos

Fuente: (*Elaboración propia)*

Para los reportes de las hojas de Inventario el proceso cambió un poco porque esta información se encontraba en la base de datos de InvIT la cual solo se podía acceder utilizando SQLAlchemy, no se podía hacer utilizando la forma de consultas tradicionales. Se debía poseer los ORM de la tabla a la que se deseaba acceder para poder utilizar las funciones de SQLAlchemy.

El backend recibía una solicitud *"post"* con un cuerpo de petición que contenía 3 campos obligatorios, estos campos eran: el tipo de hoja de inventario solicitada y un rango compuesto por dos fechas. Luego se aplicaban los filtros para devolver al frontend solamente las páginas solicitadas dentro del rango correspondiente.

En la Ilustración 40 se puede ver la consulta SQLAlchemy para el reporte de hojas de Inventario, se puede ver la función *"join"* que es la encargada de cruzar tablas uniéndolas por un campo en común y la función *"add_columns"* encargada de seleccionar las columnas de la consulta que se deseaban usar.

```
results = ImageModel.query.join(SignatureSheetModel, ImageModel.id_signature == SignatureSheetModel.id_signature, full=True).join(
    TypeModel, TypeModel.id_type == SignatureSheetModel.id_type).add_columns(
    TypeModel.name, SignatureSheetModel.id_signature, SignatureSheetModel.id_employee, SignatureSheetModel.first_name,
    SignatureSheetModel.last_name, ImageModel.image_url, SignatureSheetModel.updated)
```

Ilustración 40 - Consulta con SQLAlchemy para Reporte de hojas de InvIT

Fuente: (*Elaboración propia)*

En la Ilustración 41 se puede ver la tabla de resultados mostrada al usuario después de hacer la consulta a la base de datos.

Total Results: 75

Page Type	Page ID	Employee ID	First Name	Last Name	Image Url	Created At
Onboarding	1351	307000293	Jose	Cerrato		2020/01/06
Onboarding	707	3070011111	Pedro	Nelson		2019/01/14
Onboarding	712	307000249	Allan	Paz	712307000249	2019/01/14
Onboarding	713	307000250	Estephany	Fonseca	713307000250	2019/01/14

Ilustración 41 - Resultados del Reporte de hojas de Inventario

Fuente: (*Elaboración propia)*

A los usuarios se les permitía exportar a un archivo XLSX los resultados utilizando el botón "export" que se mostraba al final de la tabla de resultados.

4.1.10. Creación del Servicio para exportar archivos XLSX.

Una de las características que el módulo de reportes debía tener era la capacidad de guardar los resultados obtenidos en documentos que luego se podrán consultar. Para poder guardar los resultados de las consultas hechas se utilizó la librería de *"file-saver"* de Angular.

Lo primero que se debía hacer era crear el servicio de Angular para que fuera referenciado desde el componente de reportes, luego se debía enviar un arreglo con los resultados de la consulta realizada y un nombre que representará el nombre de archivo; el servicio transformaba toda esta información a un archivo .xlsx y luego se ejecutaba una función que hacía que se descargara el documento en el ordenador del usuario.

En la Ilustración 42 se puede ver el servicio que creaba y descargaba el archivo con los resultados de la consulta. En la línea 11 de la Ilustración se ve la función *"exportAsExcelFile()"* que se encargaba de crear y nombrar el archivo .xlsx, también se puede ver la función

"saveAsExcelFile()" en la línea 23, que se encargaba de guardar el archivo de Excel en la computadora del usuario.

```
8   @Injectable()
9   export class ExcelGeneratorService {
10    constructor() {}
11    public exportAsExcelFile(json: any[], excelFileName: string): void {
12      const worksheet: XLSX.WorkSheet = XLSX.utils.json_to_sheet(json);
13      const workbook: XLSX.WorkBook = {
14        Sheets: { data: worksheet },
15        SheetNames: ["data"],
16      };
17      const excelBuffer: any = XLSX.write(workbook, {
18        bookType: "xlsx",
19        type: "array",
20      });
21      this.saveAsExcelFile(excelBuffer, excelFileName);
22    }
23    private saveAsExcelFile(buffer: any, fileName: string): void {
24      const data: Blob = new Blob([buffer], { type: EXCEL_TYPE });
25      FileSaver.saveAs(
26        data,
27        fileName + "_export_" + new Date().getTime() + EXCEL_EXTENSION
28      );
29    }
30  }
31
```

Ilustración 42 - Servicio de creación de archivos .xlsx

Fuente: (*Elaboración propia)*

En la Ilustración 43 se muestra como se ve un reporte exportado como archivo XLSX.

Resource ID	Resource Name	Asset Tag	Serial Number	Acquisition Date	Component Name	Asset Type	State	User ID	First Name
496	-lnohn0061.loe.cc	LNOHN0061	FV5GT52	2015-08-03	Precision T1700	Workstation	In Use	6909	Dennis
532	-lnohn0054.loe.cc	LNOHN0054	FV4KT52	2015-09-02	Precision T1700	Workstation	In Use	6911	Saul
653	-lnohn0038.loe.cc	LNOHN0038	FV4HT52	2015-08-03	Precision T1700	Workstation	In Use	8109	Nohelia
1207	-lnohn0083.loe.cc	LNOHN0083	FV4JT52	2015-08-03	Precision T1700	Workstation	In Use	10204	Hector
1226	-lnohn0057.loe.cc	LNOHN0057	FV3MT52	2015-09-02	Precision T1700	Workstation	In Use	7808	Estephany
1631	lnpshn0805.loe.c	LNPSHN0805	872QY12	2014-08-22	Precision T1700	Workstation	In Use	3304	Isaias
1859	-lnohn0173.loe.cc	LNOHN0173	FV5HT52	2015-09-02	Precision T1700	Workstation	In Use	4259	Felix
2343	lnpshn0574.loe.c	LNPSHN0574	9QMJDZ1	2014-01-16	Precision T1700	Workstation	In Use	6912	Angel
2345	lnpshn0695.loe.c	lnpshn0695	1T3MQ22	2014-12-05	Precision T1700	Workstation	In Use	10202	Francisco
2353	lnpshn0554.loe.c	LNPSHN0554	9QKMDZ1	2014-02-15	Precision T1700	Workstation	In Use	9002	Fernando
2560	rpool-room.hnsc.	LNPSHN0576	9QMKDZ1	2014-01-16	Precision T1700	Workstation	In Use	931	LIVERPOOL
8406	lnpshn0566.loe.c	LNPSHN0566	9QLLDZ1	2014-02-16	Precision T1700	Workstation	In Use	6902	Oscar
513	-lnohn0023.loe.cc	LNOHN0023	587RT52	2015-09-05	OptiPlex 9030 AIC	Workstation	In Use	10801	Daniel
560	lnpshn0891.loe.c	LNPSHN0891	B61NQ22	2014-12-04	OptiPlex 9030 AIC	Workstation	In Use	8702	Milton
1037	-lnohn0045.loe.cc	LNOHN0043	58KPT52	2015-08-06	OptiPlex 9030 AIC	Workstation	In Use	6608	Jesus
1203	hn-lnohn.loe.corp	LNOHN0030	58JST52	2015-08-06	OptiPlex 9030 AIC	Workstation	In Use	11104	Eunice
1210	-lnohn0028.loe.cc	LNOHN0028	5SBNT52	2015-09-05	OptiPlex 9030 AIC	Workstation	In Use	5714	Daniela

Ilustración 43 - Resultados exportados como XLSX

Fuente: (*Elaboración propia)*

4.1.11. CREACIÓN DEL SERVICIO PARA EXPORTAR ARCHIVOS PDF.

El componente de "*custom reports*" también debía ser capaz de exportar en un archivo PDF los resultados que se obtienen al momento de realizar una consulta, esta característica se creó con la ayuda de la librería "*pdfmake-wrapper*" para Angular.

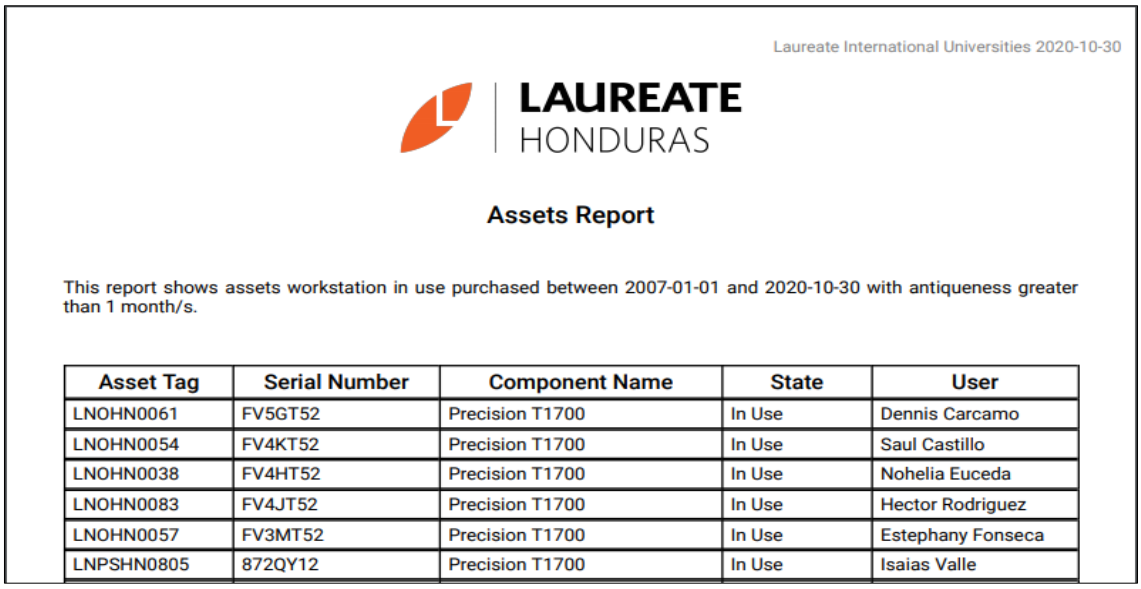
La librería anteriormente mencionada es una librería que mejora la librería "*pdfmake*", la forma de crear archivos PDF con la nueva librería es más fácil de usar y entender permitiendo al programador personalizar cada uno de los elementos que un archivo PDF pueda tener.

Fue necesario el estudio de la librería antes de poder crear el archivo, se llevaron pruebas de creación y exportación antes de poder integrar la característica funcional al componente de "*custom reports*". El documento resultante debía tener: un encabezado, el logo original de Laureate Honduras, descripción de la información que se muestra en el reporte, los resultados de la consulta realizada e información de contacto de Laureate.

También se necesitó limitar la cantidad de información que se mostraría en el PDF pues este archivo debía ser breve pero siempre mostrando los campos más relevantes que importen a la persona que realiza el reporte.

En la Ilustración 44 se puede ver el PDF creado con los resultados que mostraría una consulta de los reportes de los activos. Se observa que la información que se muestra está limitada a 5 de los 12 campos disponibles.

Para que el usuario pueda ver un reporte más detallado debe usar la opción de exportar los resultados como un archivo XLSX.



Laureate International Universities 2020-10-30

LAUREATE HONDURAS

Assets Report

This report shows assets workstation in use purchased between 2007-01-01 and 2020-10-30 with antiqueness greater than 1 month/s.

Asset Tag	Serial Number	Component Name	State	User
LNOHN0061	FV5GT52	Precision T1700	In Use	Dennis Carcamo
LNOHN0054	FV4KT52	Precision T1700	In Use	Saul Castillo
LNOHN0038	FV4HT52	Precision T1700	In Use	Nohelia Euceda
LNOHN0083	FV4JT52	Precision T1700	In Use	Hector Rodriguez
LNOHN0057	FV3MT52	Precision T1700	In Use	Estephany Fonseca
LNPSHN0805	872QY12	Precision T1700	In Use	Isaias Valle

Ilustración 44 - Resultados exportados como PDF

Fuente: (*Elaboración propia)*

4.1.12. CREACIÓN DE LA SECCIÓN DE REPORTES GUARDADOS.

Otra característica que el módulo de reportes debía tener es la opción de poder guardar un reporte para que este pueda ser consultado nuevamente si el usuario lo deseaba. El módulo debía ser capaz de listar, borrar y ejecutar los reportes guardados como favoritos para facilitar la consulta de reportes frecuentes.

En primer punto se creó la funcionalidad de poder guardar un reporte como favorito después que se le mostraran al usuario los resultados de su consulta, el usuario solamente debía ingresar el nombre con el cual quería guardarlo. Fue necesaria la modificación de la interfaz del usuario porque la versión creada inicialmente no mostraba de forma clara los reportes y no era compatible con la forma en la que esta característica se había solicitado.

Por la parte de backend el reporte guardaba 6 valores los cuales son: el nombre del reporte, el nombre del usuario que lo guardó, el tipo de reporte, la fecha de creación, los valores que el usuario estableció, y una pequeña descripción del reporte.

En la Ilustración 45 se puede ver la interfaz que el usuario ve en la sección de reportes guardados y las acciones que puede realizar con ellos.

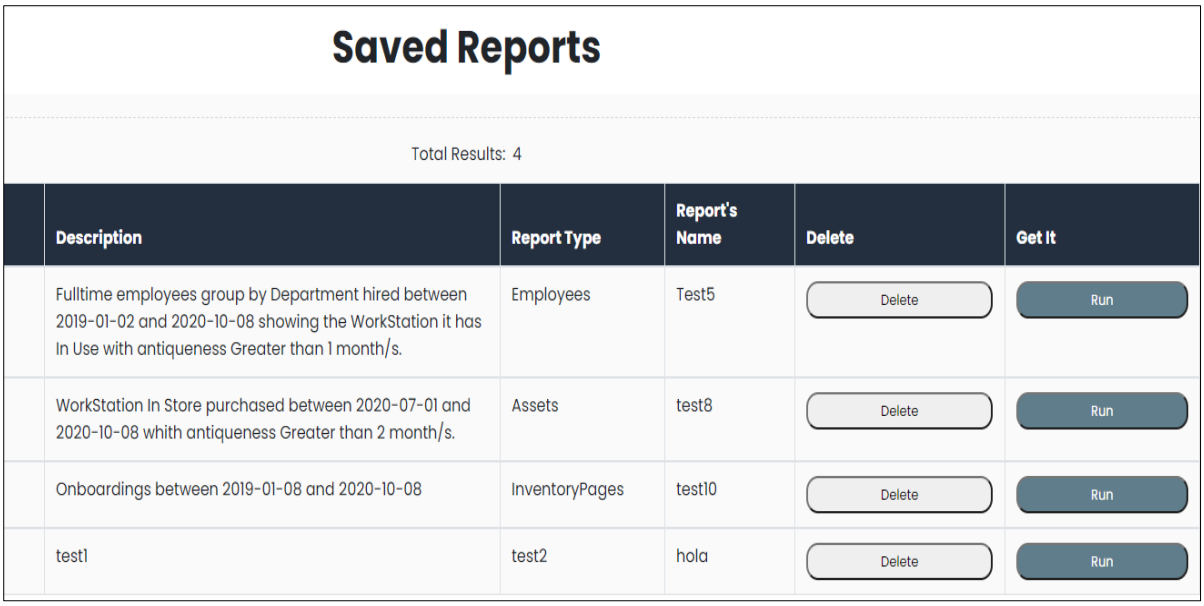


Ilustración 45 - Módulo de Reportes Guardados

Fuente: (*Elaboración propia)*

Como la información de los reportes se debía guardar en la base de datos, fue necesario que se creara la tabla en la base de datos de InvIT y su ORM correspondiente para que se pudiera consultar con SQLAlchemy. La tabla fue nombrada *"tbl_saved_reports"* y contaba con sus 6 campos descritos anteriormente, también se siguió la nomenclatura que el resto de las tablas de la base de datos usaba.

Con el módulo de *"saved-reports"* se crearon en el backend otros métodos necesarios para la correcta funcionalidad de la administración de reportes guardados, algunas de estos son:

- saveReport(): este método guardaba el reporte como favorito.
- deleteReport(): este era el encargado de borrar los reportes guardados.
- extractReportValues(): este recuperaba los parámetros y valores de los reportes para su ejecución.
- getReportDescription(): este recuperaba la descripción del reporte para ser mostrada al usuario.
- todayDate(): esta función se encargaba de devolver la fecha actual.

Un problema que se encontró fue que al guardar los valores de los reportes para posteriormente ser recuperados para la ejecución se pensó como una estructura JSON, pero la conversión del JSON como cadena de caracteres a un Objeto de Typescript fallaba por la

incorrecta estructuración del JSON, se hicieron cambios en la forma de estructurar el JSON para que la compatibilidad de JSON y Typescript se pudiera dar.

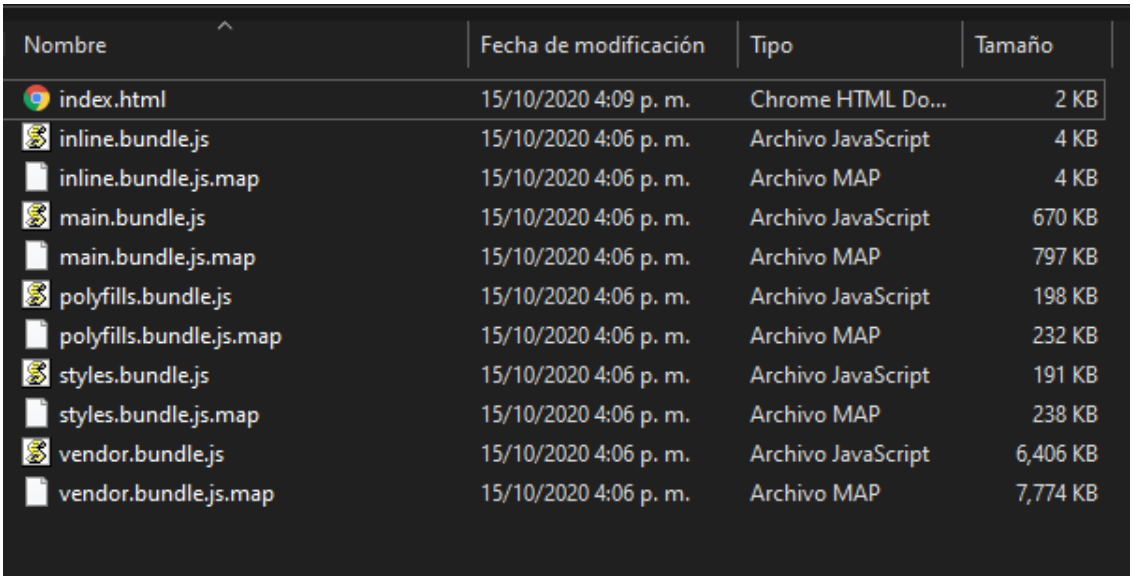
4.1.13. LANZAMIENTO A PRODUCCIÓN DE CUSTOM REPORTS V1.

El lanzamiento a producción de *"custom reports"* se hizo con la ayuda del Ing. Julio Zúniga quien es el administrador del servidor donde se aloja la aplicación de InvIT.

En primer lugar, se debía realizar un punto de restauración por si el lanzamiento de la nueva aplicación tenía errores que causen fallas totales o parciales en el sistema de InvIT.

Para el segundo paso, se debían cambiar los URL de InvIT QA a InvIT de producción, luego se necesitaba crear el minificado de la aplicación de InvIT ya con el módulo de custom reports integrado, esta acción se realizó utilizando el comando "ng build" el cual deja el Frontend listo para el lanzamiento en una carpeta del proyecto llamada "dist". Dentro de la carpeta se generan archivos "bundle.js" y un archivo "index.html".

En la Ilustración 46 se puede ver la estructura del Frontend después de ejecutar el comando "*ng build*".



Nombre	Fecha de modificación	Tipo	Tamaño
index.html	15/10/2020 4:09 p. m.	Chrome HTML Do...	2 KB
inline.bundle.js	15/10/2020 4:06 p. m.	Archivo JavaScript	4 KB
inline.bundle.js.map	15/10/2020 4:06 p. m.	Archivo MAP	4 KB
main.bundle.js	15/10/2020 4:06 p. m.	Archivo JavaScript	670 KB
main.bundle.js.map	15/10/2020 4:06 p. m.	Archivo MAP	797 KB
polyfills.bundle.js	15/10/2020 4:06 p. m.	Archivo JavaScript	198 KB
polyfills.bundle.js.map	15/10/2020 4:06 p. m.	Archivo MAP	232 KB
styles.bundle.js	15/10/2020 4:06 p. m.	Archivo JavaScript	191 KB
styles.bundle.js.map	15/10/2020 4:06 p. m.	Archivo MAP	238 KB
vendor.bundle.js	15/10/2020 4:06 p. m.	Archivo JavaScript	6,406 KB
vendor.bundle.js.map	15/10/2020 4:06 p. m.	Archivo MAP	7,774 KB

Ilustración 46 - Estructura del Frontend minificado

Fuente: (*Elaboración propia)*

En el archivo llamado "index.html" se debía hacer un cambio para que apunte a los bundles del proyecto, el cambio necesario era que se debía poner dentro de las etiquetas "<script>" el siguiente código:

```
<script type="text/javascript" src="{{url_for('static', filename='inline.bundle.js')}}"></script>

<script type="text/javascript" src="{{url_for('static', filename='polyfills.bundle.js')}}"></script>

<script type="text/javascript" src="{{url_for('static', filename='styles.bundle.js')}}"></script>

<script type="text/javascript" src="{{url_for('static', filename='vendor.bundle.js')}}"></script>

<script type="text/javascript" src="{{url_for('static', filename='main.bundle.js')}}"></script>
```

Después de que se hacen estos cambios se reemplazaron los archivos de la aplicación que esta alojada en el servidor por los archivos nuevos del proyecto, en la carpeta *"templates"* se debía reemplazar el archivo index.html por el nuevo index.html.

En la Ilustración 47 se puede ver el archivo siendo reemplazado utilizando un administrador de archivos de servidor llamado FileZilla.

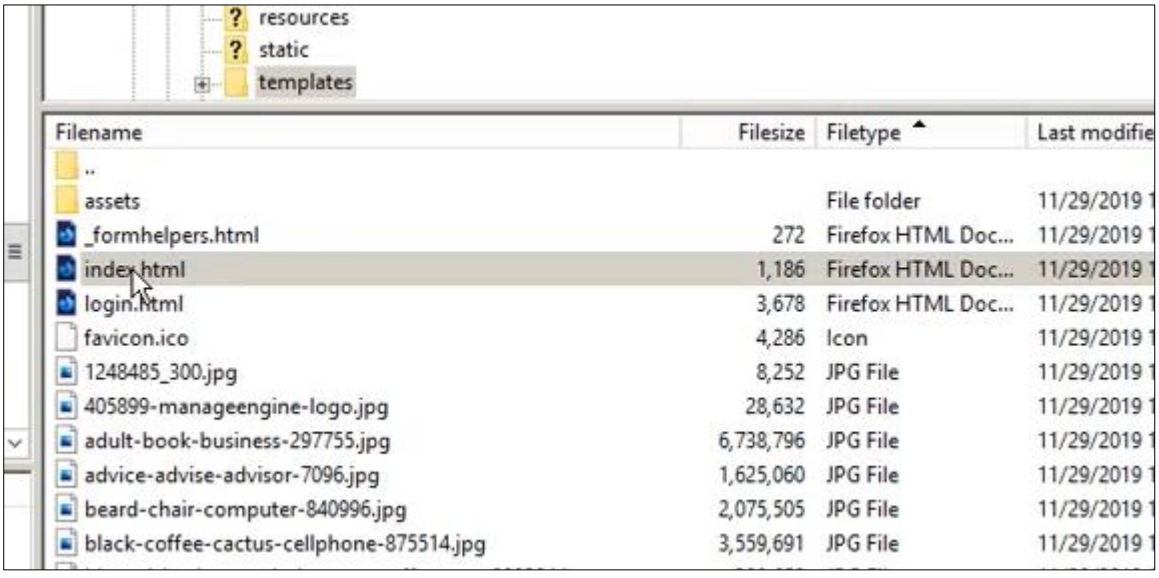


Ilustración 47 - Reemplazo de index.html

Fuente: *(FileZilla,2020)*

Como tercer paso se reemplazaban los archivos "bundle.js" que se encontraban en la carpeta *"static"* por los nuevos archivos generados por el proyecto que tiene *"custom reports"* agregados.

En la Ilustración 48 se pueden ver los archivos "bundle.js" que fueron reemplazados durante este lanzamiento.

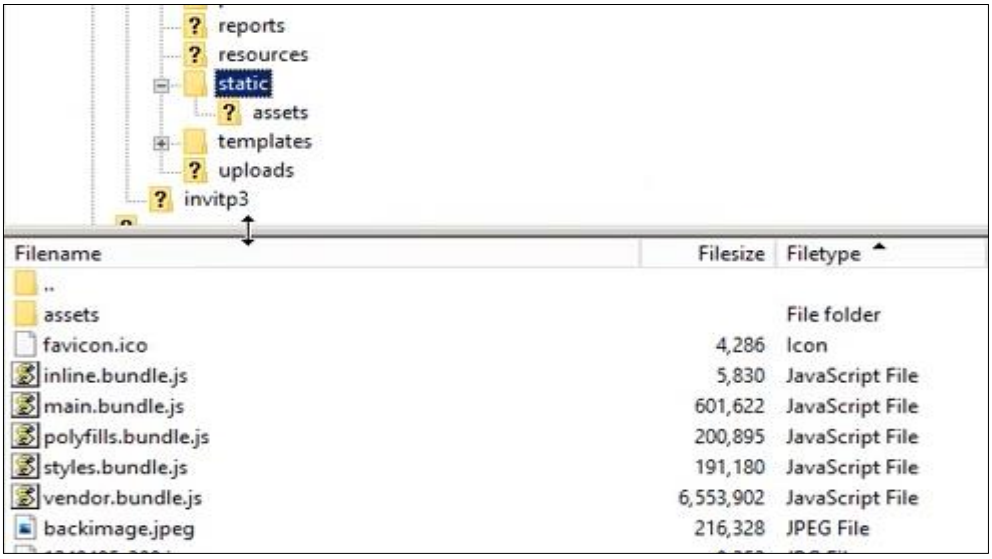


Ilustración 48 - Archivos "bundle.js" de InvIT

Fuente: *(FileZilla,2020)*

Para la parte del backend se debían reemplazar los archivos que se usaron en la elaboración del módulo de *"custom reports"* y agregar los archivos nuevos a la carpeta del backend que se encuentra en producción. El archivo que se agregó se llama "*customReports.py*" y se modificaron el app.py y el archivo de los modelos de la base de datos llamado "*signatura sheet"*, estos dos últimos debían ser reemplazados en el backend de producción.

Este procedimiento generalmente se hace automáticamente si se usaba un servidor que reflejara cada uno de los cambios que se van subiendo a la nube, InvIT se actualiza de la forma tradicional, es decir, de forma manual.

Después del lanzamiento se realizó la documentación necesaria de cada uno de las funciones y archivos creados para el Backend y Frontend.

Para hacer las pruebas de calidad y de funcionamiento se necesitó del apoyo de las personas que integran IT Operations a las cuales se les solicitó retroalimentación y sugerencias de mejora para la siguiente versión de *"custom reports"*.

Algunas de las mejoras solicitadas fueron: notificaciones con el estilo general de la aplicación, agregar el logo de Laureate al módulo, la capacidad de ordenar los resultados por cada uno de los campos de las columnas, mostrar qué campos son obligatorios, establecer campos por defecto y agregar una página de instrucciones explicando en palabras sencillas cómo usar "*custom reports*".

Toda la retroalimentación y las solicitudes de mejora se aplicaron durante los días siguientes para que ya quedaran funcionando en el segundo lanzamiento de *"custom reports"*.

4.1.14. PLANEACIÓN DE DISEÑO PARA INVIT EMAILS.

Existe en Laureate la posibilidad de pedir prestado por un tiempo limitado cualquier tipo de equipo, pero es necesario que se lleve un control sobre este equipo en préstamo, para evitar incongruencias en los reportes y pérdidas inexplicables de activos.

Se disponía de un registro digital de los equipos que estaban en modalidad de préstamo, pero la problemática a esta modalidad se encontraba en el momento de devolver el equipo prestado, muchas veces los colaboradores e incluso los agentes de IT Operations olvidaban validar la devolución del equipo o solicitar a los colaboradores la devolución correspondiente. Este problema se veía reflejado al momento de hacer la auditoría de bodega y equipo.

Es por la problemática anterior que se solicitó al desarrollador la creación de un módulo de notificaciones por correo electrónico que, de forma automática, envíe un correo electrónico recordando a los colaboradores que tienen equipo prestado para que hagan la devolución correspondiente, el correo solo será enviado si la fecha de retorno ha vencido.

La planeación del módulo de notificaciones se realizó con la ayuda del Ing. Julio Zúniga y tomando en consideración todos los puntos antes mencionados se decidió realizar de la siguiente manera:

- InvIT será la plataforma de administración del módulo, en esta administración se podrán definir los productos como devueltos y hacer excepciones en el envío de correos electrónicos.
- InvIT Emails tendrá un Backend desarrollado en Python independiente de InvIT porque la tarea que "*InvIT Emails*" hará se programará en el servidor utilizando la herramienta llamada *"task scheduler"* o el programador de tareas para que se ejecute en cierto tiempo el cual puede variar.
- En la base de datos de InvIT se agregará una tabla que tendrá los campos necesarios para que el módulo funcione, estos campos serán: el Id de la página que contiene los activos en préstamo, el id del empleado al que pertenece la página, un conteo de los correos enviados, un campo que indique si se debe hacer excepción de enviar un correo y un campo que indique si los productos ya han sido devueltos.
- Se utilizará Mandrill que es un servicio de correos gratuitos para que se hagan los envíos de los correos a los colaboradores.

En la Ilustración 49 se puede ver en un diagrama el resumen de la planificación de InvIT Emails.

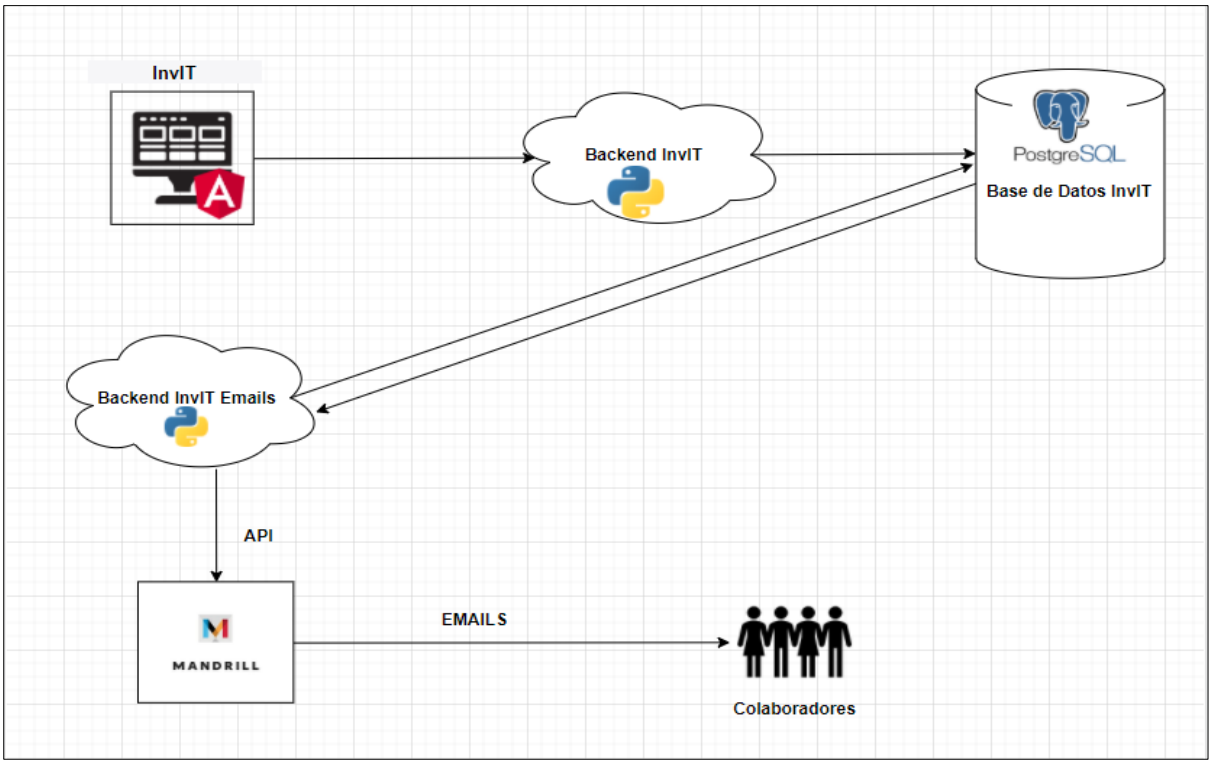


Ilustración 49 - Arquitectura de InvIT Emails

Fuente: (*Elaboración propia*)

4.1.15. Creación de Interfaz de Usuario de InvIT Emails.

Para la interfaz del usuario de InvIT Emails se tomaron en consideración los puntos que se mencionaron en la planeación, esencialmente en InvIT se debe llevar la administración de InvIT Emails, InvIT también es el encargado de las configuraciones que InvIT Emails debe manejar.

InvIT Emails es un módulo que se puede ejecutar de forma independiente a InvIT ya que fue pensado para que se ejecute de acuerdo con el tiempo establecido por el administrador usando el *"task scheduler"* o el administrador de tareas del servidor que sería la forma automática de ejecutarlo. También se podrá ejecutar de forma manual desde InvIT.

La interfaz del módulo de InvIT Emails consta de dos secciones principales: los "*loans*" o préstamos y las *"configurations"* o configuraciones.

La sección de los préstamos lista todas las páginas de préstamo existentes con la información relevante en forma de tabla, existe un botón por cada una de las páginas para que de forma rápida se pueda consultar qué productos son los que se prestaron bajo el registro de esa página, también existe la posibilidad de apagar el envío de correo electrónico a las personas en caso de que se desee crear una excepción, en la misma sección existe un botón llamado "*Send Emails Now*" que permite que los correos se envíen de forma manual en ese instante. En la Ilustración 50 se puede ver la interfaz administrativa para el envío de notificaciones por correo electrónico.

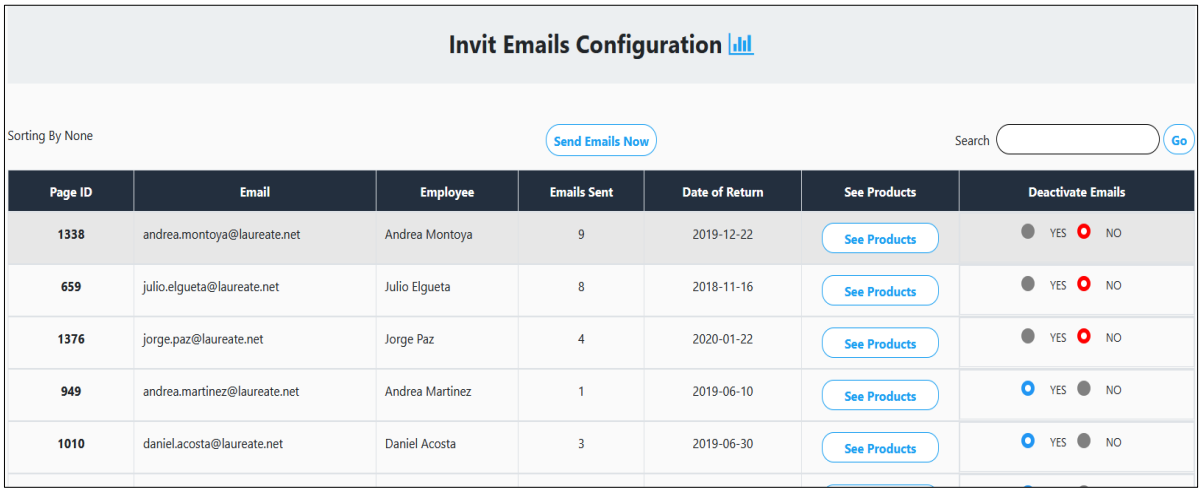


Ilustración 50 - Administración de notificaciones por correo electrónico

Fuente: (*Elaboración propia*)

En la sección de configuración para InvIT Emails se le da al administrador la posibilidad de establecer las rutas del archivo de configuración que usa el módulo y también la posibilidad de establecer la ruta para el archivo ejecutable de InvIT Emails, esto para que cuando se hagan cambios de configuración no afecte al ejecutable del mismo.

En esa sección también se podían hacer cambios de administradores, es decir, remover y agregar nuevos administradores de InvIT Emails quienes recibirían un correo resumiendo todos los correos que se envíen cada vez que se ejecute el envío de correos desde el módulo.

En la Ilustración 51 se puede ver la interfaz administrativa de las configuraciones de InvIT Emails.

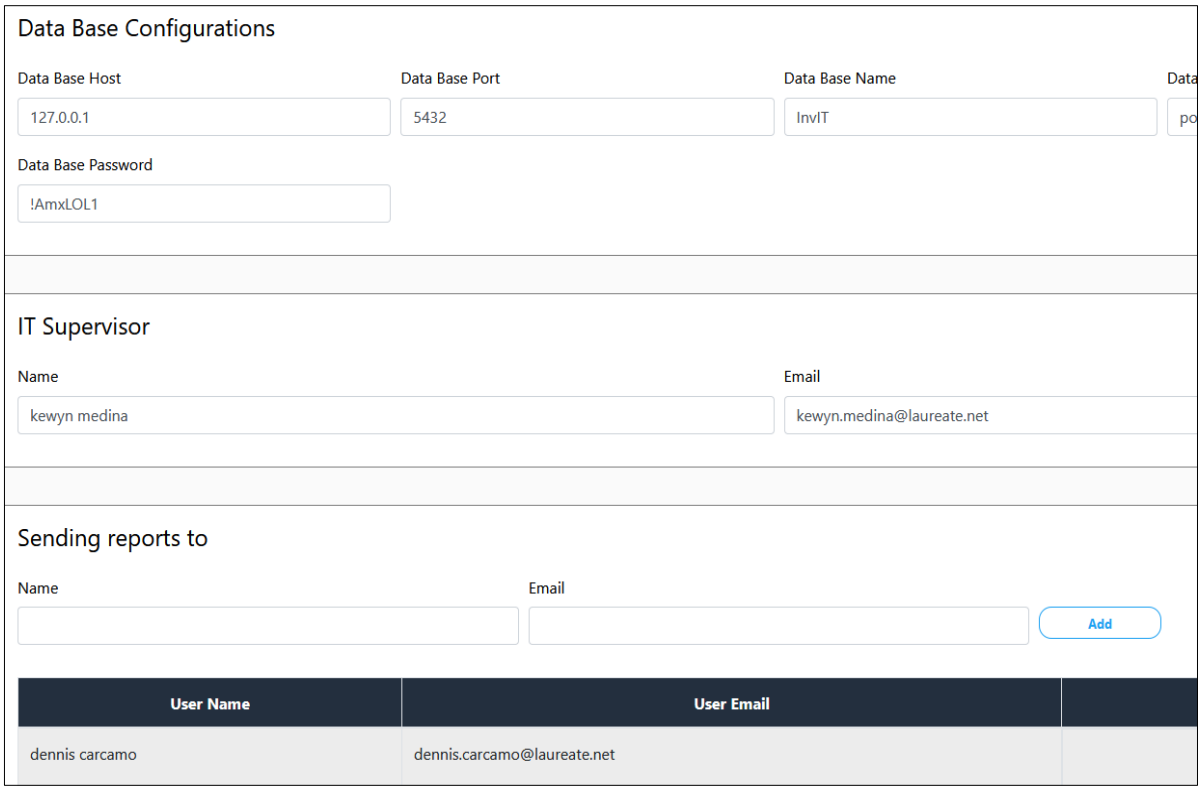


Ilustración 51 - Interfaz de configuraciones para InvIT Emails

Fuente: (*Elaboración propia*)

Las configuraciones se administran utilizando archivos de configuración con extensión ".ini" los cuales pueden ser creados y modificados con la librería "ConfigParser" de Python.

En la Ilustración 52 se pueden ver las diferentes secciones que utilizaba InvIT Emails para su configuración y ejecución.

Ilustración 52 - Estructura del archivo .ini

Fuente: (*Elaboración propia*)

4.1.16. Creación del Backend de InvIT Emails

Como primer paso se debían crear las consultas necesarias a la base de datos para obtener la información que hace posible la funcionalidad de InvIT Emails, se realizaron consultas para:

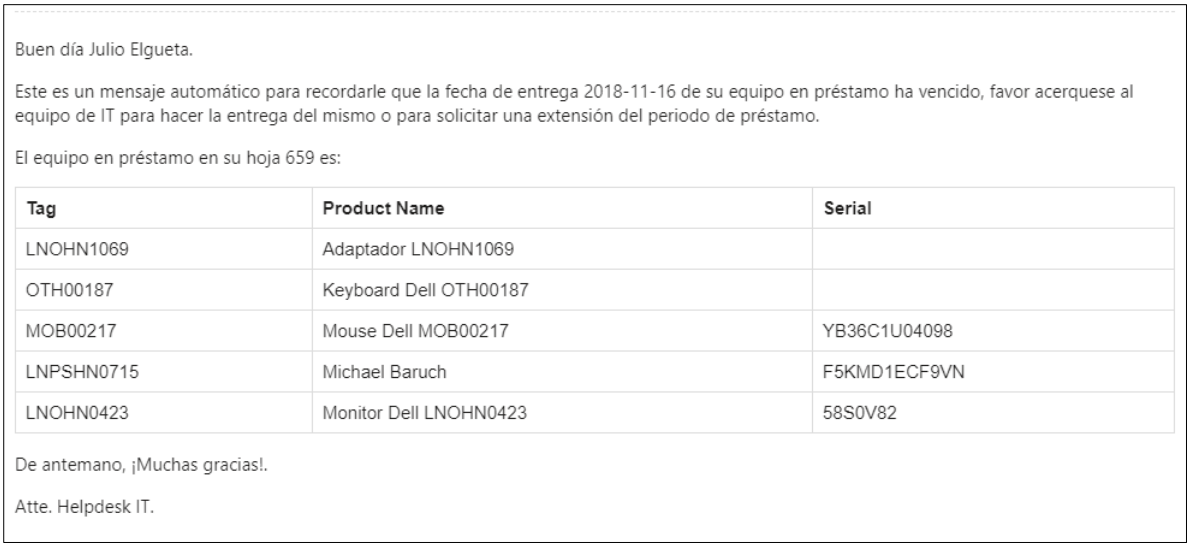
- Obtener las páginas en préstamo: esta consulta se mostraba en forma de tabla al usuario. En el Anexo 4 – Consulta SQL para recuperar las páginas con activos en préstamo, se puede ver la estructura de esta consulta.
- Obtener la información de los usuarios a los que se les debe enviar el correo: esta consulta extraía una lista de usuarios a los que se le enviarán las notificaciones por correo correspondientes. En el Anexo 5 – Consulta SQL para obtener la lista de direcciones de correos electrónicos, se puede ver la estructura de esta consulta.
- Encender y apagar el envío de correos electrónicos: esta era una de las consultas más importantes porque permitía la desactivación y activación de correos para una página en específico. En el Anexo 6 - Consulta SQL para activar y desactivar el envío de correos respectivamente, se puede ver la estructura de esta consulta.

- Obtener los productos en préstamo registrados en una página: esta consulta permitía mostrarle al usuario de forma rápida qué productos se encuentran en la página consultada.

Después de tener las consultas necesarias se debía estudiar la API especial de Mandrill para comprender la funcionalidad y poder hacer los envíos de los correos. Mandrill es un servicio de correos en el que se debe tener una cuenta para poder hacer uso de su API, Laureate ya tenía una cuenta activa en este servicio la cual se usó para el desarrollo de este módulo.

Una vez que se tenía el conocimiento necesario para el uso de la API de Mandrill se crearon los servicios que permitían enviar los correos electrónicos a los usuarios, los correos resúmenes a los administradores de IT y al supervisor de IT.

En el correo se le mencionaba al usuario de qué se trata el mensaje y se le mostraba una tabla con los activos en préstamo que poseía. En la Ilustración 53 se puede ver un ejemplo de los correos que se le enviaban al usuario.



Buen día Julio Elgueta.

Este es un mensaje automático para recordarle que la fecha de entrega 2018-11-16 de su equipo en préstamo ha vencido, favor acerquese al equipo de IT para hacer la entrega del mismo o para solicitar una extensión del periodo de préstamo.

El equipo en préstamo en su hoja 659 es:

Tag	Product Name	Serial
LNOHN1069	Adaptador LNOHN1069	
OTH00187	Keyboard Dell OTH00187	
MOB00217	Mouse Dell MOB00217	YB36C1U04098
LNPSHN0715	Michael Baruch	F5KMD1ECF9VN
LNOHN0423	Monitor Dell LNOHN0423	58S0V82

De antemano, ¡Muchas gracias!.

Atte. Helpdesk IT.

Ilustración 53 - Ejemplo de correo enviado por InvIT Emails

Fuente: (*Elaboración propia*)

4.2. Cronograma de Actividades

En la Ilustración 54 se puede ver el cronograma de actividades.

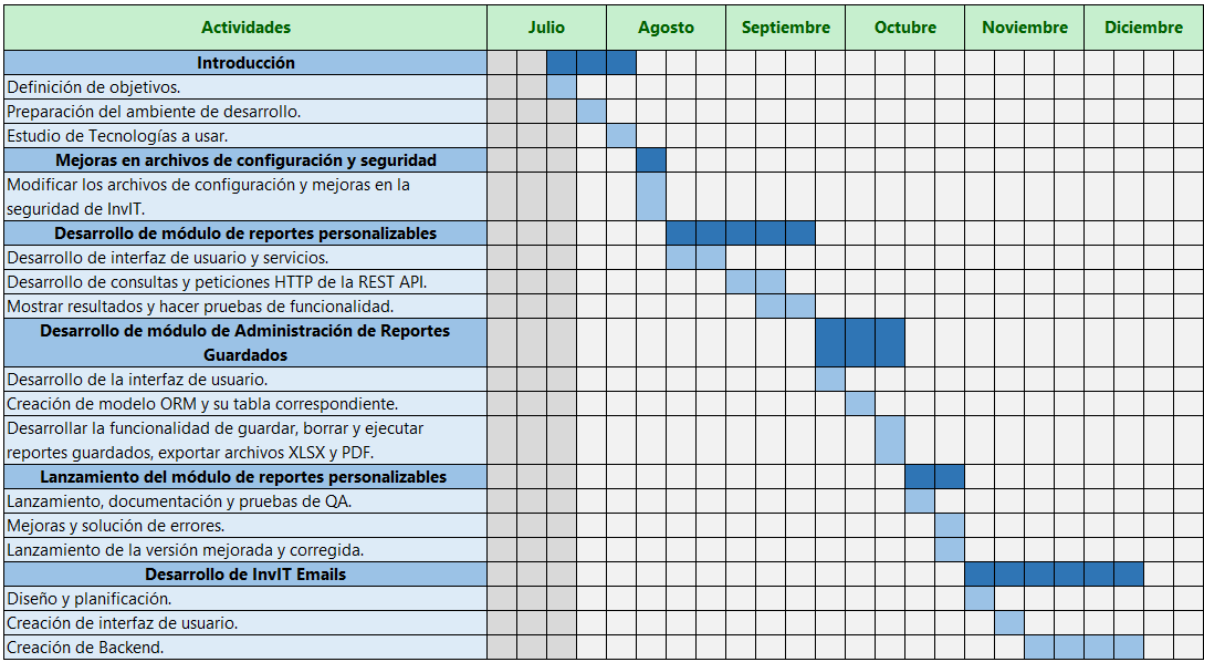


Ilustración 54 - Cronograma de Actividades.

Fuente: (*Elaboración propia)*

V. CONCLUSIONES

1. Se logró la creación e implementación del módulo de reportes personalizables en el cual el usuario puede generar reportes de empleados, páginas de inventario y activos de IT Operations. Este módulo contiene una sección de editar, borrar y volver a ejecutar reportes guardados, siempre manteniendo las restricciones de uso al personal exclusivo de IT Operations.
2. Se logró implementar la funcionalidad de poder exportar en un archivo XLSX o PDF los resultados de las consultas hechas en el módulo de reportes de forma fácil, automática y rápida que provean al usuario de "*custom reports*" un reporte oficial de Laureate IT Operations.
3. Se logró implementar un sistema de seguridad que ayude al versionamiento de la aplicación manteniendo siempre la autenticación con credenciales del dominio de Laureate, de esta forma se mantiene la confidencialidad de la información.
4. Se logró la creación e implementación de un sistema de notificaciones por correo electrónico llamado *"InvIT Emails"* que recuerda a todos los colaboradores de Laureate que tengan equipo en préstamo devolverlos una vez que hayan excedido la fecha de devolución.
5. Se logró implementar la función de poder ver y consultar los cambios y movimientos realizados a *"custom reports"* y a *"InvIT Emails"*.

VI. RECOMENDACIONES

1. Se debe disponer de diferentes ambientes, no se recomienda usar el mismo ambiente de QA para hacer las pruebas de calidad y desarrollar porque existe la posibilidad de llegar a producción con errores en la aplicación.
2. Se debe mantener una organización, documentación y separación clara de los archivos y módulos de la aplicación para la fácil comprensión de su estructura, que ayude al desarrollador al momento de trabajar en la aplicación de InvIT.
3. Se debe tener mayor participación del departamento de IT en generar los requisitos que se espera que el módulo en desarrollo tenga para darle al desarrollador una idea clara de lo que debe cumplir y las características que debe incluir.

BIBLIOGRAFÍA

1. Bayas, I. Y. G., & Martínez, M. C. (2017). La Gestión De Inventario Como Factor Estratégico En La Administración De Empresas. Negotium; Maracaibo, 13(37), 109-129.
2. Características—SQLAlchemy. (s. f.). Recuperado 30 de agosto de 2020, de https://www.sqlalchemy.org/features.html
3. Caules, C. Á. (2018, febrero 14). Angular 5 Hello World y su funcionamiento. Arquitectura Java. Recuperado el 28 de agosto de 2020, de https://www.arquitecturajava.com/angular-5-hello-world-y-su-funcionamiento/
4. Chart.js | Gráficos HTML5 de código abierto para su sitio web. (s. f.). Recuperado 30 de agosto de 2020, de https://www.chartjs.org/
5. Correa, A. (2010, octubre 10). GESTIÓN DE ALMACENES Y TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN (TIC)—ProQuest. Recuperado el 28 de agosto de 2020, de https://search.proquest.com/docview/849224391/9F2B800EA1D0431EPQ/5?accountid=35325
6. Font Awesome. (s. f.). Recuperado 30 de agosto de 2020, de https://fontawesome.com
7. Fruhlinger, J. (2020). What is TypeScript? Strongly typed JavaScript - ProQuest. InfoWorld.Com. Recuperado el 22 de agosto de 2020, de https://search.proquest.com/docview/2392125663/74AA3BFC66F74C5BPQ/1?accountid=35325
8. Guevara, V. (2016, febrero 1). Sintaxis de Programación. Verónica Guevara. Recuperado el 22 de agosto de 2020, de https://guevaracds1984.wordpress.com/sintaxis-de-programacion/
9. iainfoulds. (s. f.). Active Directory Domain Services Overview. Recuperado 31 de agosto de 2020, de https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview

10. Intro to TypeScript—DEV. (s. f.). Recuperado 30 de agosto de 2020, de https://dev.to/racheladaw/intro-to-typescript-4d8k

11. Introductory Tutorial of Python's SQLAlchemy. (2013, abril 19). Python Central. Recuperado el 28 de agosto de 2020, de https://www.pythoncentral.io/introductory-tutorial-python-sqlalchemy/

12. JavaScript. (s. f.). Documentación web de MDN. Recuperado 31 de agosto de 2020, de https://developer.mozilla.org/es/docs/Web/JavaScript

13. Kumar, J. (2018). Why Angular is the Best Framework—ProQuest. Recuperado el 28 de agosto de 2020, de https://search.proquest.com/docview/2150199788/67976D0B799A4452PQ/4?accountid=35325

14. Lau, E. (s. f.). Software de gestión de activos TI | ITAM (IT Asset Management)—ManageEngine AssetExplorer. Recuperado 29 de agosto de 2020, de https://www.manageengine.com/latam/asset-explorer/

15. Librerías para la programación web. (s. f.). Recuperado 31 de agosto de 2020, de https://aprende-web.net/librerias/

16. López S, C. A. (2009). Cómo mantener el patrón modelo vista controlador en una aplicación orientada a la WEB. Revista Inventum; Bogota, 4(7), 72-78. Recuperado el 20 de agosto de 2020, de http://dx.doi.org/10.26620/uniminuto.inventum.4.7.2009.72-78

17. Muller, M. (2005). Fundamentos de administración de inventarios. Norma.

18. Otto, M., & Thornton, J. (s. f.). Bootstrap. Recuperado 29 de agosto de 2020, de https://getbootstrap.com/

19. Pastronio, M. (2019, marzo 2). Frontend vs Backend: What's the Difference? Recuperado el 28 de agosto de 2020, de https://www.pluralsight.com/blog/software-development/front-end-vs-back-end

20. Python, R. (2018, julio 4). Crear documentos PDF en Python con ReportLab. Recursos Python. Recuperado el 26 de agosto de 2020, de https://recursospython.com/guias-y-manuales/crear-documentos-pdf-en-python-con-reportlab/

21. ¿Qué es BACKEND y FRONTEND? (Guía completa). (s. f.). EDteam - Educación con honestidad. Recuperado 30 de agosto de 2020, de https://ed.team/blog/que-es-backend-y-frontend-guia-completa

22. Qué es un diagrama entidad-relación. (s. f.). Lucidchart. Recuperado 30 de agosto de 2020, de https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion

23. ¿Qué es un framework y para qué se utiliza? | Orix Systems. (s. f.). Recuperado 31 de agosto de 2020, de https://www.orix.es/que-es-un-framework-y-para-que-se-utiliza

24. Quickstart—Flask-RESTful 0.3.8 documentation. (s. f.). Recuperado 30 de agosto de 2020, de https://flask-restful.readthedocs.io/en/latest/quickstart.html

25. Ronacher, A. (s. f.). Flask: A simple framework for building complex web applications. (1.1.2) [Python; OS Independent]. Recuperado 30 de agosto de 2020, de https://palletsprojects.com/p/flask/

26. UNITEC, Facultad de Ingeniería (2019). Manual para Redacción y Presentación de Informes Técnicos de Proyecto y Práctica Profesional (Sexta Edición). Honduras.

27. Yosemary Durán. Administración del inventario: elemento clave para la optimización de las utilidades en las empresas - Visión Gerencial (enero-junio, 2012). Recuperado el 28 de agosto de 2020, de https://www.redalyc.org/pdf/4655/465545892008.pdf

ANEXOS

ANEXO 1 - DESCRIPCIÓN DE LAS TABLAS DE LA BASE DE DATOS DE INVIT.

A continuación, se describen las tablas que forma actualmente la base de datos de InvIT.

Tabla tbl_privileges:

Almacena la información de todos los privilegios que puede tener un rol específico, y así limitar su alcance dentro del aplicativo, por aspectos de escalabilidad se realizó de esta manera ya que si el sistema sigue creciendo en el futuro solo se deben insertar los nuevos privilegios.

Sus campos son:

- id: número entero, llave primaria.
- privilege_name: texto máximo de 60 caracteres, nombre del privilegio.

Tabla tbl_rol:

Almacena los diferentes roles o tipos de usuario que puede tener el sistema al igual que la tabla tbl_privileges maneja los mismos principios de escalabilidad.

Sus campos

- id_role: número entero, llave primaria.
- role_name: texto máximo 70 caracteres, nombre del rol.

Tabla tbl_privileges_x_roles:

Almacena los distintos privilegios que puede contener un rol específico.

Sus campos:

- id role: número entero, llave foránea tbl_roles es el identificador del rol.
- id_privilege: número entero, llave foránea, tbl_privileges identificador del privilegio.

Tabla tbl_users:

Almacena los usuarios que pueden acceder al sistema:

Sus campos:

- id: número entero, llave primaria.
- id_role: número entero, llave foránea, referencia tbl_roles, identifica el rol del usuario.
- user_name: texto máximo 60 caracteres, con nombre del usuario.
- created_by: texto máximo 60 caracteres, con el nombre de usuario de quien creó el usuario nuevo.
- modified_by: texto máximo 60 caracteres, contiene el nombre de usuario que modificó la tupla.
- created_by: fecha de creación de la tupla.
- modified_at: fecha de modificación de la tupla.

Tabla tbl_type

Almacena información correspondiente al departamento de IT Operations de Laureate.

Sus campos:

- name: texto máximo 80 caracteres, se refiere al nombre del departamento.
- address: texto máximo 400 caracteres, se refiere a la dirección del departamento.
- phone: texto máximo 20 caracteres, con el número de contacto del departamento.
- terms: texto de 500 caracteres con los términos y políticas de la empresa.
- image: texto máximo 100 caracteres.
- enable: número entero pequeño para control.

Tabla tbl_signature_sheet:

Almacena toda la información correspondiente a las hojas de firmas.

Sus campos:

- id_signature: número entero, llave primaria.
- updated: Fecha de modificación o creación de la hoja.
- id_type: número entero, llave foránea, identifica el tipo de hoja.
- id_empleado: texto máximo 12 caracteres con el número de empleado del usuario.
- first_name: texto máximo 20 caracteres con el nombre del usuario.
- last_name: texto máximo 20 caracteres con el apellido del usuario.
- email: texto máximo 55 caracteres con el correo institucional del usuario.
- phone: texto máximo 55 caracteres con el número celular del usuario.
- status: número entero para control
- last: número entero para control

Tabla tbl_signature_x_product:

Almacena los activos asignados a cada hoja de inventario.

Sus campos:

- id: número: entero, llave primaria.
- id_producto: texto de máximo 25 caracteres, con el código del activo.
- ciid: número entero identificador del producto en la base de datos AssetExplorer.
- id_signature: número entero, llave foránea tbl_signature_sheet, referencia a la hoja que se encuentra asignado el activo.
- product_name: texto máximo 65 caracteres, con el nombre del activo.
- serial_number: texto máximo 35 caracteres, con el número serial del activo.
- model: texto máximo 45 caracteres con el modelo del activo.

ANEXO 2 - CONSULTA PARA LOS REPORTES DE LOS EMPLEADOS.

A continuación, se muestra la consulta creada para la obtención de información para los reportes de los empleados.

```
SELECT * FROM

    (SELECT s5.userid, s5.employeeid, s5.firstname, s5.lastname,s5.createdtime,
    s5.deptname, s5.jobtitle FROM

        (SELECT s4.userid, s4.employeeid, s4.jobtitle, s4.firstname, s4.lastname,
        s4.createdtime, s3.deptname FROM

            (SELECT * FROM

                (SELECT s1.userid, s1.employeeid, s1.jobtitle, s1.status,

                s1.firstname, s1.middlename, s1.lastname, contract.createdtime

                FROM public.sduser AS s1

            LEFT JOIN Aaauser AS contract

            ON s1.userid = contract.user_id) m1

        WHERE m1.jobtitle NOT LIKE '%Intern%' AND m1.jobtitle NOT LIKE
        '%Practicum%' AND  m1.jobtitle NOT LIKE '%Contractor%' AND m1.jobtitle
        NOT LIKE  '%Null%' AND m1.jobtitle <> '') s4

    LEFT JOIN

        (SELECT d1.userid, d2.deptname

        FROM userdepartment AS d1

        LEFT JOIN departmentdefinition AS d2

        ON d2.deptid = d1.deptid) s3

    ON (s3.userid = s4.userid)) s5
```

```
        LEFT JOIN resourceowner as ro
        ON ro.userid = s5.userid)users
LEFT JOIN
(SELECT * FROM
        (SELECT ct3.resourceid, ct3.resourcename, ct3.assettag, ct3.acquisitiondate,
        ct3.componentname, ct3.componenttypename, RT.displaystate FROM
                (SELECT ct1.resourceid, ct1.resourcename, ct1.assettag, ct1.acquisitiondate,
                ct1.resourcestateid, ct1.componentname, ct2.componenttypename  FROM
                        (SELECT r1.resourceid, r1.resourcename,r1.assettag, r1.acquisitiondate,
                        r1.resourcestateid, r2.componentname, r2.componenttypeid
                        FROM resources AS r1
                        LEFT JOIN componentdefinition AS r2
                        ON r1.componentid=r2.componentid) ct1
                LEFT JOIN componenttype AS ct2
                ON ct1.componenttypeid = ct2.componenttypeid) ct3
        LEFT JOIN resourcestate AS RT
        ON RT.resourcestateid = ct3.resourcestateid) re
LEFT JOIN
        (SELECT rw.resourceid, rw.userid
        FROM resourceowner AS rw ) rr
        ON rr.resourceid = re.resourceid)items
ON users.userid = items.userid
ORDER BY deptname
```

ANEXO 3 - CONSULTA SQL PARA LOS REPORTES DE LOS ACTIVOS.

A continuación, se muestra la consulta creada para la obtención de información para los reportes de los activos.

```
SELECT sheet_sheetType.id_employee, sheet_sheetType.id_signature,
sheet_sheetType.first_name, sheet_sheetType.last_name, sheet_sheetType.name FROM

        (SELECT sheet.id_employee, sheet.id_signature, sheet.first_name,
        sheet.last_name, sheet.email,  sheet.id_type, sheetType.name

            FROM (SELECT id_employee, id_signature, first_name, last_name, email,
            status, last, id_type

            FROM tbl_signature_sheet) sheet

        LEFT JOIN tbl_type AS sheetType

        ON sheet.id_type = sheetType.id_type) sheet_sheetType

LEFT JOIN tbl_images AS images

ON sheet_sheetType.id_signature = images.id_signature
```

ANEXO 4 - CONSULTA SQL PARA RECUPERAR LAS PÁGINAS CON ACTIVOS EN PRÉSTAMO.

A continuación, se muestra la consulta SQL que se utiliza para obtener los activos en préstamo por cada página correspondiente y poder llenar la información en InvIT Emails.

```
SELECT prod_page.id, sig.email, prod_page.id_product, prod_page.ciid,
prod_page.id_signature, prod_page.product_name, prod_page.serial_number,
prod_page.model,prod_page.received_by, prod_page.email_exception, prod_page.returned,
prod_page.emails_sent, sig.updated, sig.first_name, sig.last_name
    FROM (
        SELECT prod.id, prod.id_product, prod.ciid, prod.id_signature,
        prod.product_name, prod.serial_number, prod.model, sign.received_by,
        sign.email_exception, sign.returned, sign.emails_sent
        FROM (
            SELECT received_by, email_exception, returned, emails_sent,
            id_signature FROM tbl_invit_emails) sign
        LEFT JOIN tbl_signature_x_product AS prod
        ON sign.id_signature = prod.id_signature) prod_page
LEFT JOIN tbl_signature_sheet AS sig
ON sig.id_signature = prod_page.id_signature
```

ANEXO 5 – CONSULTA SQL PARA OBTENER LA LISTA DE DIRECCIONES DE CORREOS ELECTRÓNICOS.

A continuación, se muestra la consulta SQL que genera una lista de correos electrónicos que representan los usuarios que tiene activos en préstamo.

```
SELECT email_2.id, email_2.received_by, email_2.email_exception, email_2.returned,
email_2.emails_sent, email_2.id_signature, sheet_1.email, sheet_1.first_name,
sheet_1.last_name, sheet_1.updated
FROM (
        SELECT id, received_by, email_exception, returned, emails_sent, id_signature
                FROM public.tbl_invit_emails) email_2
LEFT JOIN tbl_signature_sheet as sheet_1
ON sheet_1.id_signature = email_2.id_signature
```

ANEXO 6 - CONSULTA SQL PARA ACTIVAR Y DESACTIVAR EL ENVÍO DE CORREOS RESPECTIVAMENTE.

A continuación, se muestran las consultas que se ejecutan cuando se activa y desactiva el envío de correos electrónicos a una página de préstamo correspondiente.

```
UPDATE public.tbl_invit_emails

SET email_exception = 1

WHERE id_signature = page_id
```

```
UPDATE public.tbl_invit_emails

SET email_exception = 0

WHERE id_signature = page_id
```