UNIVERSIDAD SANTO TOMAS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
T U N J A

TÍTULO PASANTÍA

Sistema de gestión de partes aeronáuticas para la empresa IRT

PROPONENTE
Miguel Angel Avila Torres
C.C. 1049658786
Código 2263575

DIRECTOR
Henry Alfonso Guío Ávila

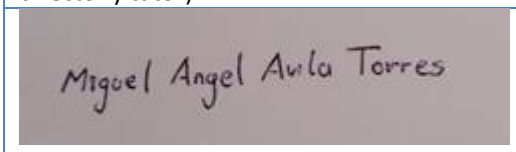
Tunja
Fecha de presentación (24, junio, 2022)

CONTENIDO

1. FICHA TÉCNICA DE LA PASANTÍA

Título	Sistema de gestión de partes aeronáuticas para la empresa IRT
Nombre Estudiante	Miguel Angel Avila Torres
Documento estudiante	1049658786
Correo electrónico	miguel.avilat@usantoto.edu.co
Director	Henry Alfonso Guio Ávila
Entidad o sector de la empresa	International Repair Technologies LLC
Lugar de ejecución de la pasantía	Florida, Estados Unidos (Remota desde Tunja, Boyacá)
Duración	4 meses

Los abajo firmantes confirman que todos los datos incluidos en la presente propuesta son correctos y verídicos, que no incumplen ninguna ley o norma vigente. (Incluir nombres y firmas de estudiantes, director y tutor).



Firma del Autor
Miguel Ángel Ávila Torres

Firma del director
Henry Alfonso Guio Ávila

2. PLANTEAMIENTO DEL PROBLEMA

Las estaciones de reparación son empresas las cuales ofrecen soluciones de bajo costo para componentes de aviación comercial o militar. En el caso de IRT (International Repair Technologies LLC) se trabaja bajo el concepto de órdenes de trabajo sobre componentes de aviación. Se ingresan las órdenes de trabajo, se utilizan procedimientos autorizados por la FAA (Federal Aviation Administration) y se aplican las técnicas de reparación autorizadas. La empresa ha de llevar control sobre el inventario de partes para reparación, partes usadas en reparaciones, kits fabricados in house, kits comprados, proveedores y clientes, así como de sus empleados.

El programa que se utiliza actualmente en la empresa es un aplicativo de Visual Basic junto con Crystal Reports elaborado en la década de los 80, sin documentación, con carencia de relaciones entre la mayoría de entidades en la base de datos e incongruencias entre la información que se muestra y almacena. Sumado a lo anterior no existe la posibilidad de re-sizear sus elementos ya que estos tienen unas dimensiones fijas ya establecidas.

La aplicación ha sido parcialmente adaptada a las necesidades de IRT y posee registros que no se utilizan, así mismo, la eficacia del software es alta (se tiene medianamente lo necesario), sin embargo, hay problemas en cuanto a la eficiencia, ya que este producto es complicado de manejar y varias operaciones automatizables han de ser realizadas manualmente por los operadores.

En síntesis, el problema consiste en que el software no se adapta completamente a las necesidades de la empresa, es complicado de manejar, desaprovecha espacio de almacenamiento, consume mucho tiempo y concentración por parte de los usuarios.

3. OBJETIVOS

3.1. Objetivo General

Desarrollar una beta de software para gestionar inventario, personal, manuales y ordenes de trabajo en la empresa IRT, utilizando arquitectura orientada a servicios y una variante personalizada de Extreme Programming como metodología de desarrollo.

3.2. Objetivos Específicos

1. Desarrollar el documento de especificación de requisitos, el diagrama de clases y los diagramas de casos de uso por medio del diagrama de la base de datos de la empresa IRT, previa aprobación de los stakeholders.

2. Diseñar la herramienta informática para establecer las vistas de usuario que permitan analizar el prototipo de software a desarrollar, utilizando el modelo de referencia “4+1” propuesto por Kruchten.

3. Implementar el prototipo del sistema creando el modelo físico de la base de datos en PostgreSQL 13, el backend y la REST API con Django REST Framework y Django ORM, el frontend con Angular 12.

4. Verificar la implementación en cuanto a la validación de los formularios respecto a la información que circula por estos, por medio de use case testing según ISTQB Foundation Level - Black-Box Test Techniques.

4. DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS

4.1. Actividades realizadas en el proyecto

Descripción de la actividad	Fecha inicio	Fecha entrega	Producto(s) entregado(s)
Diseño Vista lógica	Febrero 1	Febrero 4	Modelo ER Base de datos y de clases
Diseño Vista de desarrollo	Febrero 4	Febrero 11	Diagramas de paquetes y componentes de frontend y back-end. Especificación UML 2.5.1
Diseño Escenarios	Febrero 11	Febrero 15	Diagramas de casos de uso para los módulos de la aplicación. Especificación UML 2.5.1
Diseño Vista de Procesos	Febrero 15	Febrero 18	Diagramas de actividad para los módulos de la aplicación. Especificación UML 2.5.1
Vista de despliegue	Febrero 18	Febrero 19	Diagrama de despliegue. Especificación UML 2.5.1
Diseño Especificación de Requerimientos	Febrero 20	Febrero 28	Análisis de Requerimientos del Sistema, Especificación de Requerimientos de Software, Estructura de Desglose de Trabajo.
Pruebas de base de datos	Marzo 1	Marzo 11	Implementación de la base de datos. DDL en dialecto PostgreSQL 13
Optimización de base de datos	Marzo 11	Marzo 14	
Implementación definitiva de la base de datos	Marzo 14	Marzo 16	
Mapeo de objeto-relacional de las tablas en el backend	Marzo 17	Abril 11	Commits en el repositorio de backend, e informes a stakeholders
Creación de las rutas de la API	Marzo 23	Abril 11	
Configuración CORS	Abril 1	Abril 11	
Creación de los componentes gráficos en el frontend	Abril 11	Abril 28	Commits en el repositorio de frontend, e informes a stakeholders
Implementación lógica y mapeos en clases TypeScript	Abril 11	Mayo 2	
Conexión del frontend con la API	Abril 27	Mayo 20	
Pruebas Black-Box sobre los formularios de la aplicación	Mayo 20	Mayo 23	Application Testing 13-ésimo Commit en el repositorio de backend 19-ésimo commit en el repositorio de frontend

Tabla 1. Actividades realizadas. Autor

4.1.1. Detalles del desarrollo de las actividades

4.1.1.1 Consideraciones y diseño para la arquitectura de software

Las consideraciones para el diseño de la arquitectura de software incluyeron: el modelo de negocio, las tecnologías para el desarrollo, la optimización de la base de datos, el diseño de la interfaz y el tipo de arquitectura.

En la etapa de definición de las tecnologías, se consideró el uso de un backend basado en Go, Java o Python. Se eligió Python ya que este lenguaje tiene una gran "codebase" destinada a machine learning, big data e inteligencia artificial, por lo cual, si surgieran requerimientos de análisis de big data o generación de reportes estadísticos, evento que es bastante probable, Python es la tecnología más conveniente.

Ya que Python es uno de los lenguajes más usados en la actualidad y con una buena cantidad de desarrolladores, la continuación del proyecto no encontraría mayor dificultad en la búsqueda de personal capacitado para el mantenimiento y escalamiento del backend y adicionalmente los costos para ello se verían disminuidos gracias a la oferta de mano de obra.

Para la selección del framework de backend se consideraron Flask, Django, Spring, y GIN (Basado en GO). Se eligió Django + Django REST Framework (DRF) + CORS, ya que esta estrategia nos ofrece ORMs, autenticación y autorización, entre otros beneficios, junto con la flexibilidad de una REST API, una velocidad de desarrollo superior gracias a los serializadores y model view sets dentro de DRF y una capa adicional de seguridad gracias a la política CORS (Cross Origin Resource Sharing).

Respecto a la base de datos, se consideraron sólo sistemas de gestión de bases de datos (DBMS) opensource por costos futuros a la empresa. Se tuvieron en cuenta PostgreSQL, MySQL, MariaDB y MongoDB. Finalmente, por cuestiones de fiabilidad, rendimiento y conocimiento sobre el sistema, se seleccionó PostgreSQL ya que cuenta con una extensa documentación y una gran cantidad de desarrolladores que lo utilizan.

Como frameworks de frontend se consideraron React, Vue y Angular. Se seleccionó Angular teniendo en cuenta que es desarrollado y mantenido por Google, además, cuenta con muy pocas dependencias y están bien curadas, es de tipo SPA (Single Page Application) y está enfocado en velocidad y rendimiento. Angular se centra en el desarrollo de componentes de interfaz por medio de clases y está escrito en TypeScript, esto hace que la cohesión del código sea mayor y exista una mejor estandarización, comparado con React y Vue, al momento de la implementación, mantenimiento y análisis de la “codebase”.

Se decidió añadir Angular Material ya que es una librería para componentes UI la cual posee completa integración con Angular, alta calidad, es versátil y además es desarrollada y mantenida por Google.

Para los protocolos de autenticación y autorización en el frontend, se consideró el uso de JWT, teniendo en cuenta que, en las más recientes actualizaciones de revistas dedicadas a seguridad informática, se han hallado fallas de seguridad en el uso de Cookies de Sesión.

Para lograr realizar un producto de alta calidad y gran fiabilidad, se consideró fundamental el diseño de la base de datos como una herramienta que permitiera entender las necesidades reales del negocio y al mismo tiempo, un medio con el que los stakeholders pudieran realizar aportes, correcciones y sugerencias mientras se trabajaba el diseño del modelo. Con la implementación de esta técnica se logró conseguir una base de datos bastante curada que cubre las necesidades de la empresa IRT.

Posterior al diseño del modelo ER de la base de datos se procedió con la creación del diagrama de clases, el diseño de la vista de desarrollo mediante diagramas de paquetes y componentes, de los escenarios mediante diagramas de casos de uso, de la vista de procesos usando diagramas de actividades y de la vista de despliegue usando los diagramas respectivos. Todo esto bajo la especificación 2.5.1 UML del OMG.

Una vez se estableció el modelo de vista arquitectónica, se crearon los documentos de análisis de requerimientos del sistema, la especificación de requisitos de software y el Work

Breakdown Structure (WBS) que sirve como guía para entender el proceso de desarrollo que se aplicó durante la implementación del software.

Para establecer la apariencia física del producto a desarrollar, se crearon desde cero las interfaces gráficas para la aplicación web. El proceso de diseño se orientó con la técnica de UX conocida como “Paseo Cognitivo”, dado que el software es uno de dominio, los usuarios están familiarizados con la terminología, por lo tanto, al crear las interfaces como una secuencia lógica de pasos al realizar el trabajo manualmente, se simplifica el uso del producto y la curva de aprendizaje resulta menor.

Finalmente, para reducir la cantidad de pasos por parte de los usuarios, se estipuló conveniente crear las interfaces como abstracciones de actividades del negocio, siendo estas las siguientes: órdenes de trabajo, inventario, kits, personal, manuales, proveedores, clientes y misceláneos. Actividades que resumen el flujo de trabajo en la empresa.

4.1.1.2 Implementación de base de datos

Establecida la arquitectura de software y la especificación de requisitos, se implementó el modelo ER de base de datos en PostgreSQL por medio de Django ORM (herramienta incluida en el framework Django) con el fin de reducir tiempo en cuanto a la creación de mapeos objeto-relacionales con la base de datos y retrasos por incompatibilidad entre tipos nativos de PostgreSQL y tipos manejables por Django.

Se hicieron pruebas sobre la inserción de registros y las consultas en la base de datos con el fin de generar una estrategia de optimización. Con el desarrollo de esta fase se lograron consultas más óptimas y un diseño de base de datos más sólido.

Para conseguir información de prueba, se creó de manera independiente a este proyecto la herramienta de poblado de bases de datos Epsilon Databasepopulator, que consiste de una serie de scripts en Python con la capacidad de leer DDLs para SQL de dialecto PostgreSQL, describiendo la estructura, relaciones, tipo de dato y patrones requeridos sobre los campos de las tablas. Con esta herramienta se generaron suficientes registros en

diferentes ocasiones cuando se deseaba corregir el modelo o testear nuevas ideas en cuanto a optimización.

4.1.1.3 Desarrollo asíncrono de backend y frontend

El desarrollo de la aplicación frontend con Angular y Angular Material se inició con el propósito de mostrar a los stakeholders la apariencia del software para lograr un acuerdo en cuanto a la capa de presentación. Se crearon 8 módulos, 1 por cada actividad y dentro de estos se declararon los componentes correspondientes a cada actividad, sólo exportando el componente raíz del módulo.

Finalizada la creación de los componentes gráficos se procedió a realizar las primeras implementaciones del backend con Django REST Framework, mediante la creación de los viewsets para los modelos y los serializadores JSON de estos.

Posteriormente, se continuó el desarrollo en el frontend, creando interfaces para “mapear” la información de los objetos JSON recibidos como respuesta a las peticiones del frontend. Se crearon los servicios y se comenzó comprobando el correcto funcionamiento del backend trayendo información para los filtros de datos, para que, una vez seleccionado un ítem, se realizara la carga de la información del objeto y sus relaciones correspondientes.

Consecuentemente, se implementaron las funcionalidades de creación, actualización y borrado sobre las entidades en la base de datos y se procedió a la integración de estas funcionalidades con el frontend.

Finalizada esta etapa se procedió a añadir funcionalidades como notificaciones de sistema, confirmaciones de guardado de cambios y diálogos de confirmación para el borrado de información.

Finalmente se pasó a la etapa de testeo, donde se encontraron y corrigieron errores inesperados tanto en frontend y backend. Una vez terminada esta etapa, se agregó la documentación correspondiente al código fuente y a los README.MD de los proyectos de frontend y backend, finalizando de manera exitosa el desarrollo de la pasantía.

Los detalles técnicos sobre el frontend y el backend son:

Frontend

- +30 componentes
- +30 interfaces de mapeo
- +140 funcionalidades
- 8 servicios, 1 por cada actividad
- 1 servicio para almacenamiento de información de forma parcial
- 1 servicio con métodos genéricos para abstracción de peticiones al backend
- 8 módulos, 1 por cada actividad
- 1 módulo compartido para importes y centralización del código
- Un script para refactorización el cual abstrae la mayor parte de las operaciones conectando a backend y mostrando notificaciones
- Uso de:
  - Cadena de responsabilidad
  - Patrón padre a hijo
  - Inversión del control
  - Estilos globales en vez de estilos específicos de cada componente

Backend

- Uso de:
  - django
  - django-cors-headers
  - djangorestframework
  - psycopg2
- Implementación de
  - +30 modelos
  - +30 serializadores
  - +30 model view sets
  - +130 funcionalidades

Es importante anotar, que para la realización de esta pasantía se escribieron más de 10000 líneas de código.

4.1.1.4 Reuniones y entrega de avances

Por cuestiones de disponibilidad de la persona a cargo del pasante, en este caso el CEO de la empresa, las reuniones realizadas incluyeron actualizaciones de estado notorias y generalmente centradas en la apariencia y funcionalidad del software. En estas reuniones hubo pocos “change requests” sobre elementos del sistema o la interfaz.

4.1.2. Evidencias

4.1.2.1 Imágenes del software legacy

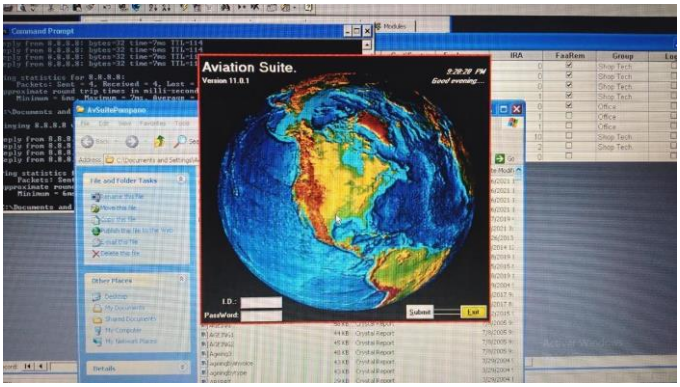


Figura 1. Interface de entrada – Software Legacy

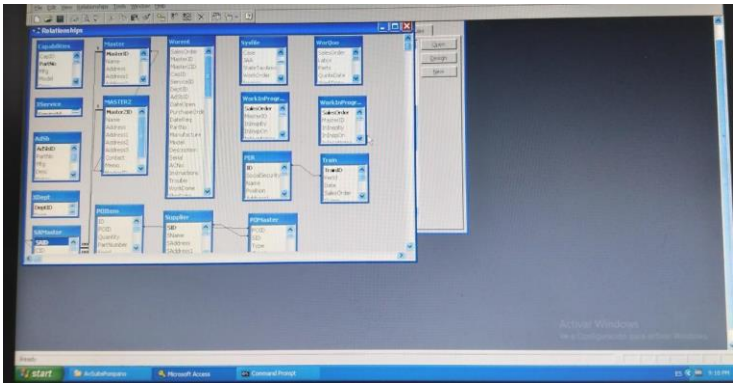


Figura 2. Visualización Base de Datos con MS Access – Software Legacy

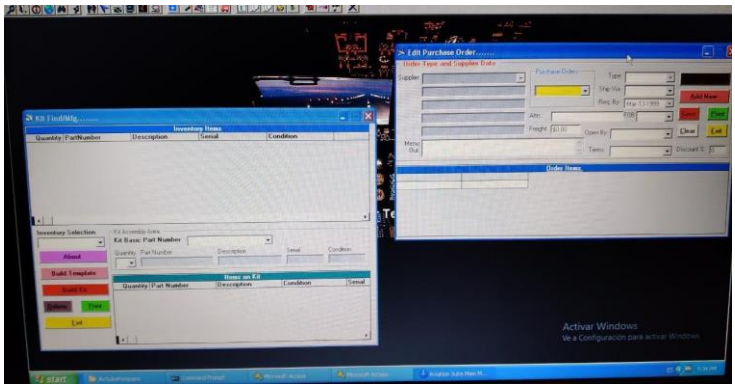


Figura 3. Proceso de Gestión de Kits – Software Legacy

4.1.2.2 Preview iteraciones del modelo ER

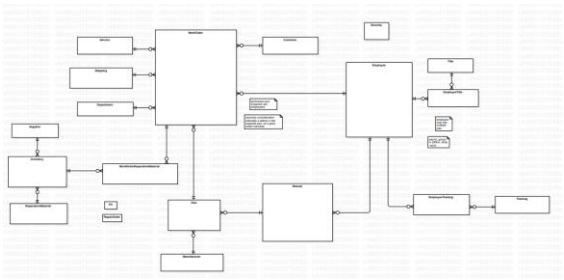


Figura 4. Iteración #1 – Modelo ER

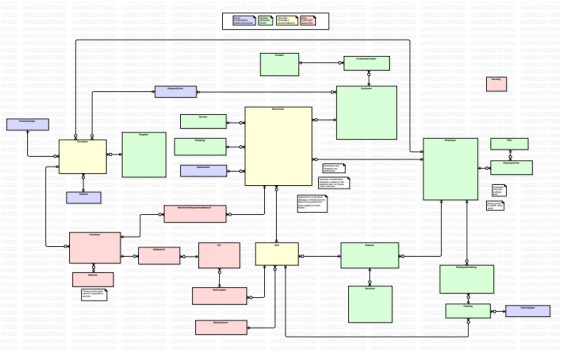


Figura 5. Iteración #2 – Modelo ER

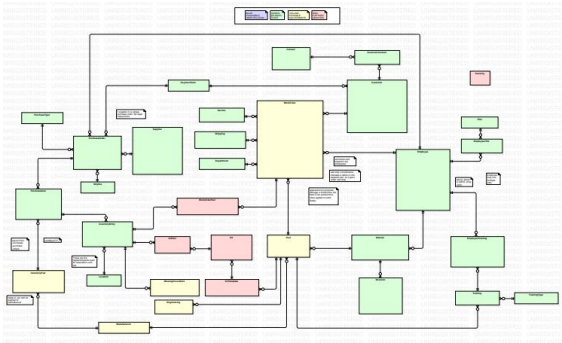


Figura 6. Iteración #3 – Modelo ER

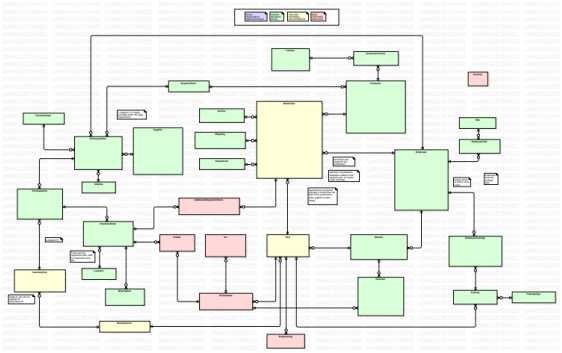


Figura 7. Iteración #4 – Modelo ER

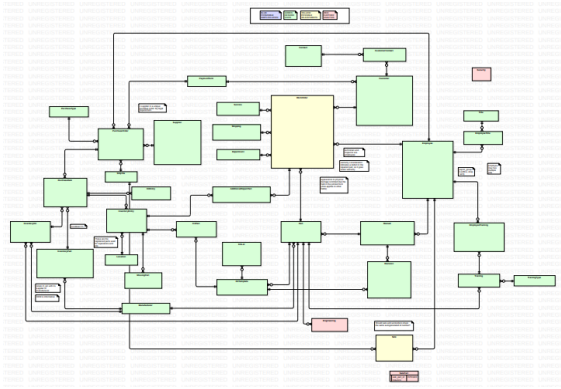


Figura 8. Iteración #5 – Modelo ER

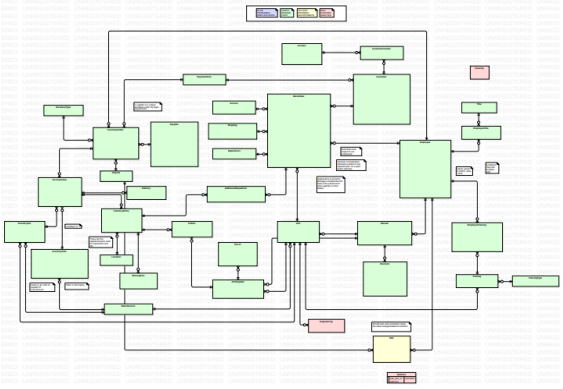


Figura 9. Iteración #6 – Modelo ER

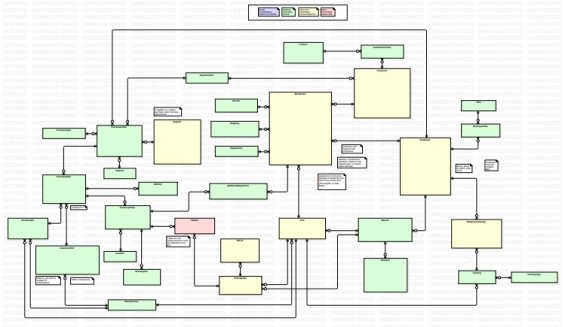


Figura 10. Iteración #7 – Modelo ER

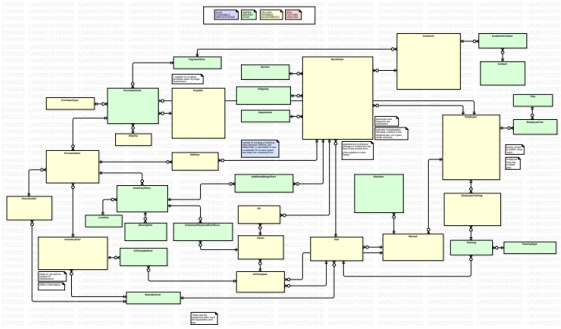


Figura 11. Iteración #8 – Modelo ER

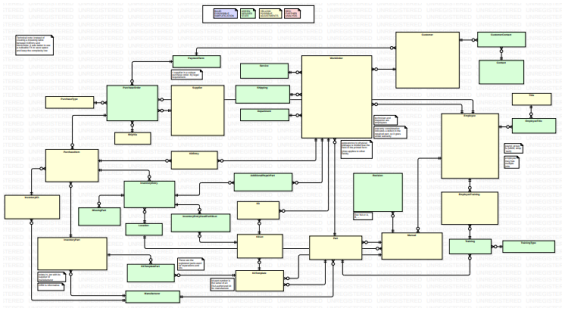


Figura 12. Iteración #9 – Modelo ER

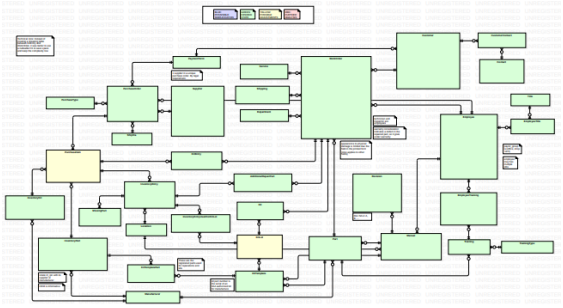


Figura 13. Iteración #10 – Modelo ER

4.1.2.3 *Imágenes de los diseños de interfaz para el software desarrollado*

Figura 14. Diseños de las Interfaces – Actividades 1 a la 3

Figura 15. Diseños de las Interfaces – Actividades 4 a 5

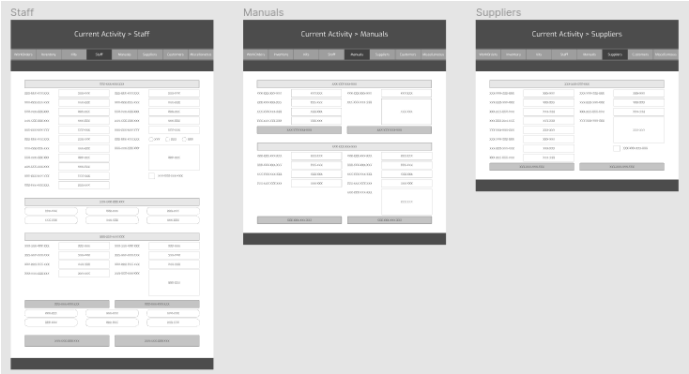


Figura 16. Diseños de las Interfaces – Actividades 6 a 8

4.1.2.4 Previews del software desarrollado

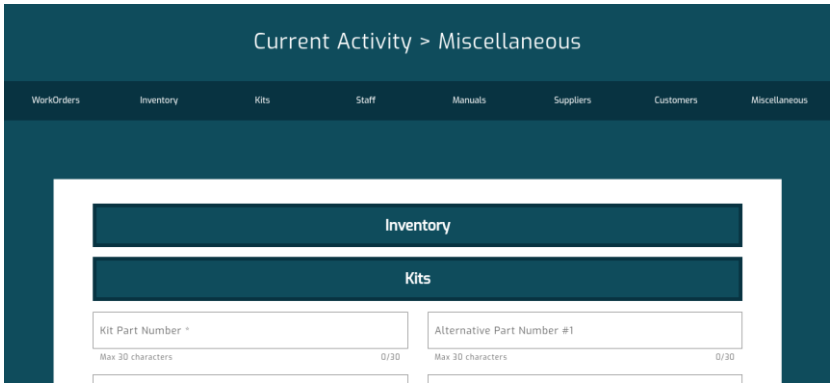


Figura 17. Preview de la actividad Misceláneos - No 1

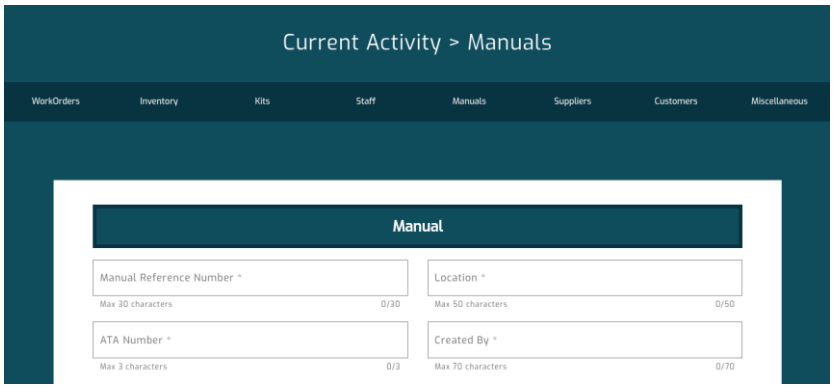


Figura 18. Preview de la actividad Manuales

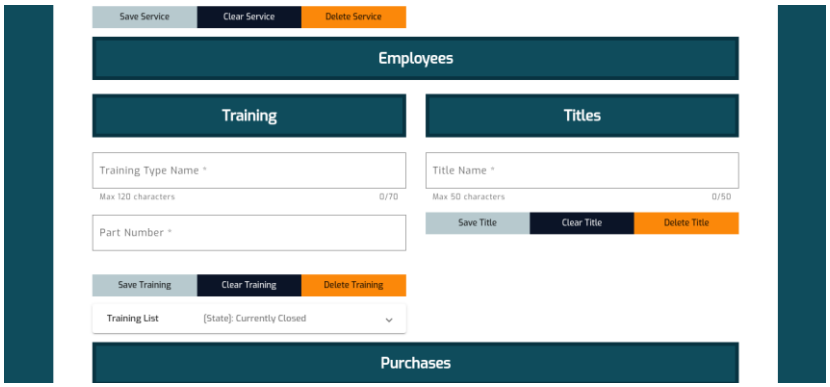


Figura 19. Preview de la Actividad Misceláneos - No 2

4.1.2.5 Listado de commits

Commit en proyecto backend	Commit en proyecto frontend
update v0.1 (cd0640d)	update v0.1 (ddb4c70)
update v0.7 (2d2bd3e)	update v0.2 (30cd459)
update v0.7 (9401174)	update v0.2 (c473df1)
delete rest_backend/core_app/migrations directory (cd5564d)	update v0.2.1 (658e1f7)
update v0.7.1 (5b89bc2)	update v0.2.2 (9dcf22c)
update v0.8.1 (8d8bbe1)	update readme (09e17a9)
update v0.8.2 (17eb5c4)	update v0.2.3 (3740a0a)
update v0.8.3 (5cfc969)	update v0.2.4 (a08f899)
update v0.8.5 (2d02349)	update v0.2.5 (cb683d7)
update v0.8.7 (dfacd8a)	update v0.2.6 (4f66471)
update v0.8.8 (4d2d5f6)	update v0.2.6 (3fe96ad)
update v0.9.0 (fa6625f)	update v0.3 (98e48d7)
update v0.9.9 beta (32c560d)	update v0.3.2 (425cd17)
	update v0.3.3 (9a3731a)
	update README (ce2bde1)
	update v0.9.1 (5cac418)
	update v0.9.2 (2cf8ff2)
	update v0.9.3 (f2be8fd)
	update v0.9.9 beta (bff3c34)

Tabla 2. Commits en repositorios backend y frontend. Autor

4.2. Herramientas utilizadas en el desarrollo del proyecto

Herramienta utilizada	Finalidad
PyCharm Professional	Desarrollo del proyecto de frontend y backend así como de Epsilon Database Populator

PSQL (PostgreSQL 13 Command Line Interface)	Inserción vía SQL script de datos de prueba en la base de datos, comprobación de schemas.
YFiles (JetBrains Extension)	Visualización de la base de datos usando radial layout para una más fácil comprobación de las relaciones en la implementación
StarUML 5.0.1	Elaboración del modelo de vista arquitectónica
Epsilon Database Populator v1.0	Generación de datos para el DDL generado por las migraciones de Django

Tabla 3. Herramientas de desarrollo. Autor

5. CONCLUSIONES

1. El trabajo de un desarrollador Full Stack es bastante extenso y ha de cubrir casi todas las áreas del desarrollo, sistemas de gestión de bases de datos, tecnologías de frontend, tecnologías de backend, integración de dependencias, conceptos claves de los frameworks utilizados en el desarrollo como lo son (IoC) inversión del control, mapeos, métodos de optimización en la carga de información, etc.
2. El trabajo de un arquitecto de software está orientado hacia la gestión del proyecto, tiene un perfil un tanto similar al de un project manager (PM), sin embargo, su trabajo se centra en crear un modelo de solución con el cual tanto los stakeholders como los desarrolladores estén de acuerdo, por su simpleza y completitud.
3. Los objetivos propuestos que lograron alcanzarse fueron:
    a. Los documentos de especificación de requisitos
    b. El diagrama de clases
    c. Los diagramas de casos de uso
    d. El diagrama de la base de datos de la empresa IRT
    e. La arquitectura de software por medio del modelo 4+1 de Kruchten
    f. La implementación de la base de datos en PostgreSQL
    g. La implementación backend con Angular y Angular Material
    h. La implementación frontend con Django y Django REST Framework
    i. El testeo según ISTQB Foundation Level - Black-Box Test Techniques

    Tal y como se planteó en los objetivos.
4. Hubo una gran dificultad para la realización de las entregas debido al poco tiempo, ya que los objetivos planteados fueron muy ambiciosos para un solo desarrollador terminando estudios universitarios. Sin embargo, se finalizó satisfactoriamente el desarrollo del software, y con este se adquirió una gran cantidad de experiencia.

6. ANEXOS

6.1. Contrato de Confidencialidad

CONTRATO DE CONFIDENCIALIDAD

Contrato de confidencialidad celebrado en la fecha **Junio 10, 2021**, **International RepairTechnologies, LLC** localizado en **1888 NW 21st Street, Pompano Beach, FL 33069 USA** y **Miguel Ángel Ávila Torres** localizado en **Calle 6 No 15 – 34, Tunja** _ al tenor de las declaraciones y clausulas siguientes:

DECLARACIONES

Declara la Empresa **International Repair Technologies, LLC (IRT)**, por conducto de sus representantes: Que es su voluntad obligarse en los términos de éste contrato. Declara **Miguel Ángel Ávila Torres**, en representación propia, que es su voluntad obligarse en los términos de éste contrato:

Declaran las partes, por conducto de sus representantes:

1. Que han decidido transmitirse mutuamente cierta información confidencial, propiedad de cada una de ellas, relacionada con tecnologías, planes de negocios internos, y otros tipos, a la que en lo sucesivo se le denominará "Información Confidencial", relativa a la relación de negocios entre las partes.
2. Que cualquiera de ellas, en virtud de la naturaleza de éste contrato, podrá constituirse como parte receptora o parte divulgante.
3. Que se reconocen mutuamente la personalidad con la que comparecen a celebrar el presente convenio y manifiestan su libre voluntad para obligarse en los términos de las siguientes:

CLAUSULAS

PRIMERA. Las partes se obligan a no divulgar a terceras partes, la "Información Confidencial", que reciban de la otra, y a darle a dicha información el mismo tratamiento que le darían a la información confidencial de su propiedad.

Para efectos del presente convenio "Información Confidencial" comprende toda la información divulgada por cualesquiera de las partes ya sea en forma oral, visual, escrita, grabada en medios magnéticos o en cualquier otra forma tangible y que se encuentre claramente marcada como tal al ser entregada a la parte receptora.

SEGUNDA. La parte receptora se obliga a mantener de manera confidencial la "Información Confidencial" que reciba de la parte divulgante y a no darla a una tercera parte diferente de sus abogados y asesores que tengan la necesidad de conocer dicha información para los propósitos autorizados en la Cláusula Sexta de éste convenio, y quienes deberán estar de acuerdo en mantener de manera confidencial dicha información.

TERCERA. La parte receptora se obliga a no divulgar la "Información Confidencial" a terceros, sin el previo consentimiento por escrito de la parte divulgante.

CUARTA. La parte receptora se obliga a tomar las precauciones necesarias y apropiadas para mantener como confidencial la "Información Confidencial" propiedad de la otra parte, incluyendo, mas no limitando, el informar a sus empleados que la manejen, que dicha información es confidencial y que no deberá ser divulgada a terceras partes.

QUINTA. La parte receptora está de acuerdo en que la "Información Confidencial" que reciba de la otra parte es y seguirá siendo propiedad de ésta última, a usar dicha información únicamente de la manera y para los propósitos autorizados en la Cláusula Sexta de este contrato y que este instrumento no otorga, de manera expresa o implícita, derecho intelectual o de propiedad alguno, incluyendo, mas no limitando, Licencias de uso respecto de la "Información Confidencial"

SEXTA. La parte receptora se obliga a utilizar la "Información Confidencial" de cualquiera de las otras partes, únicamente para la compra de una de ellas de la empresa y/o marcas relativas a servicios y productos objeto del contrato

6.1.1.1 Elementos cubiertos por el contrato de confidencialidad

Los siguientes ítems no pueden ser revelados al público debido a la información sensible que contienen sobre el modelo de negocio de la empresa y la forma en la que opera:

- Análisis de Requerimientos del Sistema
- Especificación de Requerimientos de Software
- Estructura de Desglose de Trabajo (WBS)
- Modelo de Vista Arquitectónica
    - Modelo ER
    - Diagrama de Clases
    - Diagramas de Paquetes
    - Diagramas de Componentes
    - Diagramas de Actividades
    - Diagramas de Casos de Uso
    - Diagramas de Despliegue
- Manual de Usuario
- Proyecto de Backend + Documentación
- Proyecto de Frontend + Documentación
- Diseños de la interface grafica
- Software funcional operando en condiciones standard