UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de control y rastreo de productos de importación

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Ivan Yaotzin Velázquez Rocha

DIRECTOR DE TESIS

Ing. Román V. Osorio Comparán

Ciudad Universitaria, Cd. Mx., 2017

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Índice temático

Dedicatoria

A mi madre.

Porque sin ella no habría llegado hasta este punto de mi vida.
Que, con sus valores, consejos, paciencia y -sobre todo- amor logró sacar lo mejor de mí.

Objetivo

El presente trabajo tiene como principal objetivo el mostrar de manera detallada la forma en que se implementa un sistema de software web de gestión de pedidos de una empresa. Los puntos que se tocarán, son:

- Análisis del problema.
- Diseño de la solución.
    - A nivel tecnológico.
    - A nivel base de datos.
    - A nivel requerimientos de usuario y casos de uso.
    - A nivel paradigmas de programación y modelo de desarrollo.
- Metodología a utilizar a lo largo del desarrollo.
- Desarrollo de la aplicación.
    - Configuración del ambiente de desarrollo.
    - Programación de las clases Dominio y querys SQL.
    - Programación de las clases Controlador y Servicios.
    - Creación de las vistas responsivas.
- Puesta en producción de la aplicación.
- Detalle del flujo de trabajo de la aplicación como solución tecnológica.

Como segundo objetivo es mostrar el funcionamiento de este sistema de control y rastreo, en el cual se pretende unificar la información sobre el estado en el que se encuentra un producto que ha sido pedido de exportación para luego recibirlo y enviarlo al cliente.

Con este sistema se pretende automatizar los procesos dentro de la empresa, los cuales tienen que ver con el rastreo y seguimiento de los pedidos. Además de mejorar la atención al cliente.

Introducción

La tesis **Sistema de control y rastreo de productos de importación** habla sobre el diseño y la implementación de una aplicación web, la cual es un sistema que gestiona el proceso que siguen los productos dentro de una empresa de importación. Dicho sistema es llamado TRACKING.

El sistema TRACKING está dirigido a las empresas que proveen productos de importación a cierto nicho de mercado y que, dado su modelo de negocio, venden sólo bajo pedido; es decir sólo adquieren el producto en el extranjero una vez que un cliente lo ha pedido. Dentro de los productos del ramo de ésta industria entran, por ejemplo, material profesional de áreas como medicina, meteorología, geología, etc.

El sistema TRACKING busca dar solución al problema de controlar el ciclo de vida de las órdenes de pedidos durante todo el proceso interno de la empresa.

En el presente documento se verá claramente cómo se diseña y desarrolla el sistema TRACKING de acuerdo a las necesidades generales de la industria mencionada.

Capítulo 1. Marco teórico

Este capítulo está destinado a hablar sobre los conceptos, tecnologías y procedimientos que se utilizarán en el desarrollo del sistema TRACKING.

1.1 Conceptos teóricos

Aplicación web

En ingeniería de software se describe como una serie de herramientas de software que se usa para acceder a un servidor web a través de internet o una intranet utilizando cualquier navegador. No está limitada a hacer procesamiento de información, sin embargo, su filosofía es que la información que presenta al usuario es la que se obtiene de peticiones y recepciones con el servidor.

Las principales ventajas de una aplicación web sobre las que son locales, son:

- Problemas de compatibilidad mínimos o inexistentes.
- Consume muy poca memoria y no consume espacio en el disco duro.
- Multiplataforma. Es independiente al Sistema Operativo (SO) de la computadora.
- No se afecta por virus locales, puesto que la información está protegida en el servidor.

Los principales inconvenientes son:

- Ofrecen una interfaz menos interactiva que una aplicación local.
- El procesamiento de información es más lento que en una aplicación local.

Arquitectura de software

Es un conjunto de patrones y abstracciones que permite definir la estructura, funcionamiento, interacción e infraestructura de un sistema de software al más alto nivel. Para definir todas las características y componentes de un sistema, la arquitectura se basa en requerimientos funcionales, de mantenibilidad, flexibilidad, interacción, etc. Estos requerimientos no sólo los dictan las necesidades del cliente sino la naturaleza y limitaciones de la tecnología, procesos, o cualquier factor externo que intervenga en el desarrollo y uso del sistema.

Base de datos

Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su uso posterior. Para utilizar una base de datos se requiere un sistema gestor de base de datos (**DBMS** por sus siglas en inglés). Existen varios modelos de bases de datos. Un modelo es una forma de estructurar y acceder a la información. El modelo relacional es el más utilizado actualmente y se basa en usar “relaciones” entre entidades; donde cada entidad es la representación de un objeto o acción del sistema. Cada entidad puede apuntar a otras entidades, definiendo así su relación.

DOM

Acrónimo de **Modelo de Objetos de Documentos** (**Document Object Model**, en inglés). Es una interfaz que proporciona estándares de objetos para representar documentos HTML, XML, o cualquier texto que necesite estructurarse. W3C es un consorcio internacional que se encarga de actualizar y estandarizar DOM. El objetivo de DOM es proporcionar una estructura que permita acceder, añadir y cambiar de forma sencilla y rápida el contenido de un archivo. La mayoría de los lenguajes que se utilizan para el diseño web y actúan sobre su funcionamiento utilizan DOM; el ejemplo más representativo es JavaScript.

Framework

En ingeniería de software es una estructura conceptual (a veces también tecnológica), que da soporte para desarrollar software. Provee de la estructura de los componentes a desarrollar (basado en una arquitectura y metodología de desarrollo). Comúnmente lo que incluye son bibliotecas y lenguaje interpretado. Su principal objetivo es ofrecer alguna funcionalidad bien definida que se base en patrones de diseño. Las características de un framework son:

- Dicta el flujo del programa que se está creando.
- Tiene un comportamiento bien determinado que debe de ser útil, definido e identificable durante todo el desarrollo.
- Es extensible, es decir que cuando se quiera agregar funcionalidad no sea necesario modificar el código ya hecho.

HTTP

Acrónimo de **Hypertext Transfer Protocol** (**Protocolo de Transferencia de Hipertexto**) es un protocolo de comunicación para la trasferencia de información en internet o intranet, y está a nivel de la capa de “Aplicación” en el modelo OSI. También se desprende de aquí el protocolo HTTPS (**Hypertext Transfer Protocol Secure** o sea **Protocolo Seguro de Transferencia de Hipertexto**) que es la versión segura de HTTP.

IDE

IDE (por sus siglas en inglés: **Integrated Drive Electronics**) es un ambiente de desarrollo integrado. Es una herramienta de software que proporciona servicios integrales para facilitar el desarrollo de software. Por lo regular contiene un editor de código fuente, bibliotecas, soporte a varios lenguajes, construcción automática, autocompletado inteligente de código (IntelliSence), buscador, compilador, etc. Puede ser tan complejo como se necesite; algunos tienen soporte para plugins.

Ingeniería de software

Es una disciplina que se encarga de desarrollar, mantener y operar software; esto incluye la investigación y estudio de técnicas, métodos y tecnología para cumplir sus objetivos. Sus prácticas se respaldan (y tienen su origen) en la ingeniería, matemáticas y ciencias de la

computación. Su objetivo principal es obtener la solución más adecuada al problema según sea el caso. Aunque dentro de sus objetivos también están:

- Aumentar la calidad en el producto.
- Detectar la mejor forma de obtener tiempos y costos para el desarrollo de software.
- Generar procesos y normas para regular el desarrollo, mantenimiento y soporte del software.
- Crear y organizar equipos de trabajo efectivos.

El término de ingeniería de software es bastante general y no corresponde a un título universitario definido o profesión. Todo lo que compete a la ingeniería de software se puede definir por los procesos y recursos que se utilizan en cada una de las etapas del desarrollo o gestión de software. Al conjunto de estas etapas se le llama el ciclo de vida del software, que es:

- Toma de requerimientos.
- Análisis.
- Diseño.
- Arquitectura de la aplicación.
- Construcción.
- Pruebas.
- Producción.
- Documentación.
- Mantenimiento.
- Operación.
- Expiración.

Ya que se maneja una gran cantidad de información y procesos durante el ciclo de vida del software es necesario utilizar un modelo que lo englobe. Un modelo es un conjunto de prácticas y una filosofía que se usan para desarrollar, mantener y evolucionar software consiguiendo los mayores beneficios posibles. Los más comunes son:

- Modelo de cascada.
- Modelo de prototipos.
- Modelo en espiral.
- Modelo orientado a objetos.

Los aspectos que toma en cuenta la ingeniería de software para trabajar, son:

- Necesidades del cliente.
- Calidad del software.
- Recursos disponibles (tiempo, dinero, personas y tecnología).
- Impacto social, económico y ambiental.
- Restricciones legales.
- Ética profesional.

Lenguaje informático

Es aquel usado y asociado con el uso de computadoras. Existen diferentes tipos de lenguajes informáticos; algunos son:

- Lenguaje de marcado.
- Lenguaje de programación.
- Lenguaje declarativo.
- Protocolo de comunicación.

Lenguaje de marcado

Forma de codificación que se basa en agregar etiquetas o marcas que contienen información sobre la estructura del texto y su presentación. No es un lenguaje de programación ya que no contiene aritmética. Su objetivo es el de ayudar a presenta el texto al usuario final de manera estructurada y entendible. El más usual es XML.

Lenguaje de programación

Es un lenguaje formal (lenguaje formal es aquel que está formado por símbolos y reglas de unión formalmente especificados) que sirve para declarar procesos que puedan ser llevados a cabo por una computadora. Un lenguaje de programación es caracterizado por sus reglas sintácticas y semánticas, pues ellas definen su estructura y alcance. Por lo regular un lenguaje de programación se hace basado en algún paradigma de programación; dicho de otra forma, está diseñado para ser usado bajo cierto paradigma, aunque no está casado con uno sólo y por lo regular da la libertad para que el programador siga el paradigma que desee.

Metodología de desarrollo

En ingeniería de software es una metodología para el planificar, estructurar, implementar y controlar el desarrollo de software. Su principal objetivo es presentar técnicas que permitan desarrollar software de calidad. Su campo de trabajo es administrativo, pues se basa en utilizar de forma correcta los recursos del proyecto para alcanzar las metas deseadas en un tiempo establecido. Básicamente hay dos tipos de metodologías: ágiles y tradicionales (aunque en la realidad, ambas metodologías se mezclan.)

- Metodología ágil: Es aquella donde el modelo en el que se basa es iterativo e incremental, o derivados. Su principal característica es que se basa en equipos bien integrados, auto-organizados y multidisciplinarios con capacidad de colaborar entre sí para tomar decisiones ágiles y a corto plazo. El software al que le da prioridad no es aquel que tiene muchas o toda la funcionalidad, sino aquel que está libre de errores. Sus principales características son:

    - Flexibilidad ante los cambios del proyecto.
    - Clientes son partes activos del desarrollo.
    - Grupos pequeños en el mismo lugar físico.
    - Pocas políticas y normas.

  - Técnicas basadas en heurísticas a partir de prácticas de codificación.
  - Le dan poco peso a la documentación exhaustiva.
  - Le dan mucho peso a la retroalimentación con el cliente.

- Metodología tradicional: Es aquella donde el modelo en el que se basa es el de cascada o derivados. Buscar el mayor control a partir de la rigidez en los procesos. Sus principales características son:

  - Se basa en etapas de desarrollo bien definidas.
  - Seguimiento estricto del plan inicial de desarrollo.
  - Rigidez ante los cambios del proyecto, de manera lenta.
  - Roles y jerarquías bien definidos.
  - Reuniones con el cliente de manera formal.
  - Grupos grandes de trabajo que pueden estar distribuidos geográficamente.
  - Basada en normas y estándares de desarrollo.
  - Mucha importancia a la documentación robusta.

Modelo de desarrollo

Para ingeniería de software un modelo de desarrollo es la abstracción del proceso (o ciclo de vida) de desarrollo, mantenimiento y evolución del software, teniendo como objetivo conseguir los mayores beneficios posibles. Los más comunes son:

- Modelo de cascada: se basa en ordenar de forma rigurosa cada etapa del desarrollo donde cada una de ellas debe comenzar cuando termina la anterior. Fue el primer modelo en desarrollarse y ha sido bastante criticado; sin embargo, sigue siendo de los más utilizados en la actualidad. Su principal ventaja es el orden y simplicidad de entender el proceso. Su principal desventaja es que existe mucha dependencia y cualquier falla en el diseño tiene un gran impacto en las posteriores etapas.
- Modelo de prototipos: se basa en construir todos los módulos del sistema de manera rápida para ver el funcionamiento general; es decir primero hacer prototipos. Luego sobre esos prototipos se van agregando funcionalidad hasta obtener un sistema complejo. Su principal ventaja es que se puede ver el funcionamiento desde el inicio. Su principal desventaja es que no asegura tener al final un sistema de buena calidad u óptimo.
- Modelo en espiral: se basa en una combinación de prototipos con cascada; es decir que comienza a construir el sistema en pequeñas partes, pero respetando las etapas como en cascada, y al acabar la última etapa regresa a la primera para corregir y aumentar funcionalidad. Su principal ventaja es que es bastante controlado y mitiga mucho los posibles errores que afectan gravemente a un proyecto. Su principal desventaja es que es muy complicado calcular tiempos y costos.
- Modelo orientado a objetos: está basado en el paradigma de programación orientado a objetos (POO), por lo tanto, los conceptos en los que se basan en su análisis y diseño es en los pilares de la POO. Su principal ventaja es que facilita la reutilización de software y que su diseño fácilmente puede ser representado en diagramas UML. Su principal desventaja es que puede resultar un poco antinatural el diseño, y a veces muy complejo.

Mapeo Objeto-Relacional

El ORM (**Obect-Relational Mapping**, en inglés) es una técnica de programación que convierte los tipos de datos utilizados en POO a entidades de una base de datos relacional, y viceversa. Esto es que se está creando una base de datos orientada a objetos virtual sobre la base de datos relacional. Es una herramienta muy útil para el programador ya que este sólo se ocupa de diseñar una clase en POO la cual se verá reflejada en la BD (o viceversa), y podrá manejar de manera mucho más sencilla la persistencia y consulta de datos.

Paradigma de programación

En ingeniería de software, un paradigma de programación es un enfoque particular sobre la solución de problemas que están claramente definidos. Básicamente un paradigma se define por la forma de abstraer el problema y los elementos que lo forman, así como la manera de implementar la solución. Está en constante evolución y cambio, así como los problemas que maneja. No hay una forma concreta de estandarizar los paradigmas, sino que deben de ser evaluados y aceptados por toda la comunidad de desarrolladores. Actualmente los paradigmas con mayor uso y aceptación son:

- Programación orientada a objetos.
- Programación funcional.
- Programación declarativa.
- Programación imperativa.
- Programación orientada a aspectos.
- Programación orientada a eventos.

Plugin

Un “plug-in” o “add-on” (“complemento” en español) es una aplicación de software que agrega una nueva función muy específica a otra aplicación (más compleja). La aplicación principal ejecuta al plugin cuando sea necesario.

Programación Orientada a Objetos

Conocida como **POO** (**OOP** en inglés) es un paradigma de programación que se basa en abstraer los elementos de un problema en objetos. Donde un objeto es una parte del sistema que tiene un estado y comportamiento definido y particular. La interacción entre objetos va a poder generar tanto la funcionalidad del sistema como la transferencia de información. No es propio de ciertos lenguajes ni está peleado con otros paradigmas.

Las propiedades de un objeto son:

- Estado: es la capacidad de un objeto de cambiar a lo largo de su ciclo de vida y de acuerdo a su contexto; su estado se define a partir de los atributos que tiene.
- Comportamiento: es la capacidad de un objeto de tener funcionalidad; su comportamiento se define por lo métodos que tiene.

- Identidad: es la capacidad de un objeto de ser único; esta capacidad es la mayor parte del tiempo transparente al desarrollador y de esto se encarga el compilador o el ambiente de desarrollo.

Quizá el principal concepto que introdujo este paradigma es el de “clase”. Una clase se refiere a “una clase de objeto”, y es la definición del tipo de objeto que puede existir en el sistema; una vez definiendo una clase se pueden crear innumerables objetos de esa clase, y todos tendrán el mismo comportamiento y estado, pero una identidad única.

Bases de POO:

- Abstracción: es la capacidad de conceptualizar el problema que se quiere resolver de tal forma que se pueda ver en términos de objetos. El proceso de abstracción permite seleccionar las características sobresalientes de los elementos del sistema para poder diseñar las clases que se encargarán de representar los objetos y el sistema.
- Encapsulamiento: es la capacidad que tiene un objeto de ser una unidad independiente y por tanto que su comportamiento y estado sea transparente a otros objetos. Así tiene la capacidad de restringir el acceso a sus métodos y atributos hacia otros objetos. Esto lo que permite es generar un bajo acoplamiento en el sistema y su modelo de objetos.
- Polimorfismo: es la capacidad que tiene un objeto de poder comportarse de maneras diferentes de acuerdo al contexto en el que se encuentre. Esto aumenta flexibilidad del diseño del sistema.
- Herencia: es la capacidad que tiene un objeto de copiar el comportamiento y estado de otro objeto. Esto se hace a nivel de clase, y cuando se dice que una clase puede heredar a otra significa que copiará sus métodos y atributos. Esto lo que permite es la reutilización de código y la representación de jerarquías.

Programación declarativa

El paradigma de programación declarativa es lo contrario al imperativa, pues no dice qué hacer exactamente. La programación declarativa especifica un conjunto de condiciones y restricciones que describen el problema y cómo resolverlo: sólo se espera el resultado. Deben de existir mecanismos internos del ambiente sobre el cual se programa que puedan obtener la solución.

Open Source

Traducido como Código Abierto. Es aquel software desarrollado y distribuido libremente; es decir no es sólo el poder usar o adquirir el software de manera gratuita sino también poder modificar el código del software y lanzar una nueva versión, todo de manera legal. No se debe de confundir con las versiones o licencias de uso libre que algunas empresas de software tienen para sus productos, donde el uso de dicho producto con dicha licencia (licencia de “Comunidad”) es gratuito, pero no puede accederse al código fuente.

Patrón de diseño

En ingeniería de software un patrón de diseño es un enfoque y método que le da solución a tipos de problemas comunes en el desarrollo del software. Para que un patrón de diseño sea considerado como tal debe de cumplir dos principios:

- Comprobable que haya resuelto un problema real.
- Que pueda utilizarse para el mismo tipo de problema en un contexto diferente.

En general los patrones de diseño tienen como objetivos:

- Generar una colección de soluciones bien definidas y reusables al diseño de software.
- Aumentar la calidad del software utilizando soluciones bien hechas ya probadas.
- Estandarizar la forma de diseñar software.
- Crear un vocabulario formal en el diseño de software.

Tipos de patrones:

- Creacionales: aquellos relacionados con la creación de instancias (encapsulamiento y polimorfismo).
- Estructurales: aquellos relacionados con agregar nuevas clases u objetos (escalabilidad).
- Comportamiento: aquellos relacionados al comportamiento y relación que hay entre las clases y objetos.

Servidor web

También conocido como servidor HTTP. Es una herramienta de software que está siempre a la espera de peticiones de información de clientes (la aplicación web); al recibir peticiones hacen procesamiento de información y ejecución de código de aplicaciones (llamados “scripts”) para luego devolver el resultado solicitado. Se usa generalmente el protocolo HTTP para la comunicación entre el servidor y el cliente.

Sistema de gestión de base de datos

Un **SGBD** o **DBMS** (por sus siglas en inglés de **Data Base Management System**) es un conjunto de herramientas de software que permiten almacenar, acceder y extraer información de una base de datos. Proporcionan herramientas para añadir, borrar, modificar y analizar la información.

Script

Es un programa que se almacena en un solo archivo de texto. Por lo general muy sencillo y es interpretado. Sus tareas por lo regular tienen que ver con combinación de componentes, interacción con el sistema operativo o con el usuario. Son muy populares a nivel de sistema operativo y en el diseño web.

Software

Es el conjunto de componentes lógicos que componen a un sistema informático y hacen posible la realización de tareas específicas. No existe un delimitador de lo que es o implica el software, siendo que un sistema informático se puede componer de otros varios sistemas más pequeños completamente funcionales e independientes.

La IEEE lo define como:
"Es el conjunto de programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación."

SQL

Es el acrónimo de **Structured Query Language**; es un lenguaje declarativo (véase *programación declarativa*) que se usa para acceder y operar a las bases de datos relacionales.

De entre sus características más destacadas están:

- Lenguaje de definición de datos: puede definir esquemas de relación y modificación de datos.
- Lenguaje de modificación de datos: su lenguaje de consulta está basado tanto en álgebra relacional (lenguaje de consulta que describe paso a paso cómo obtener una consulta deseada) tanto como en cálculo relacional (lenguaje de consulta que describe la respuesta que se desea obtener, pero sin especificar cómo obtenerla).
- Se pueden especificar las restricciones de integridad de los datos.
- Se pueden especificar los permisos de acceso.

1.2 Tecnología utilizada

Para el desarrollo de este sistema utilicé software libre (open source), así como software comercial en versiones de prueba o gratuitas.

AJAX

JavaScript Asíncrono y XML (**Asyncronus JavaScript And XML** en inglés) es una técnica para el desarrollo de aplicaciones web interactivas sin usar un exceso de recursos. Esto lo logra haciendo que las aplicaciones se ejecuten en el cliente (en el navegador) mientras se mantiene en una comunicación asíncrona con el servidor en segundo plano; lo que logra al fin de cuentas es que no requiera recargar toda la página web cuando sea necesario actualizar sólo una parte de ella; con esto logra una gran velocidad aumentando la interactividad y experiencia de usuario.

La comunicación asíncrona en segundo plano se refiere a que la comunicación del cliente con el servidor va a ser transparente para el usuario y la ejecución de la aplicación será independiente a la respuesta del servidor (es decir, el navegador no va a esperar la respuesta del servidor para seguir trabajando); por lo tanto, todo el tiempo se estará comunicando el navegador con el servidor, intercambiando información y no va a interferir con la visualización ni comportamiento de la página.

Ajax está basada en estándares abiertos por tanto es compatible con múltiples plataformas. Combina las siguientes tecnologías:

- HTML y CSS para el diseño de la información.
- DOM (**Document Object Model**, en inglés) y JavaScript para la interacción con la página.
- XMLHttpRequest como canal de comunicación.
- XML, HTML y JSON para transferencia de datos.

Ventajas:

- Uso moderado de recursos.
- Velocidad en las aplicaciones web.
- Buena experiencia de usuario.

Desventajas:

- Si no se tiene cuidado se sobrecargará al servidor, al hacer varias peticiones a la vez.

CSS

Hoja de estilos en cascada (**Cascading Style Sheets**, en inglés) es un lenguaje que sirve para crear la presentación de un documento estructurado que está en lenguaje HTML o XML. Su filosofía es la de separar la estructura del documento y su presentación.
Dentro de sus ventajas tenemos:

- Centralizar el control de la presentación.
- Reutilización de código.
- Separar la estructura del documento de su presentación.

Algunas de sus desventajas:

- Es difícil lograr una estructura de tablas.
- No contiene ninguna función de cálculo numérico.
- No todo el código CSS puede ser controlado desde el navegador.

Grails

Es un framework libre para desarrollar aplicaciones web, desarrollado sobre el lenguaje de programación Groovy.

Los objetivos de Grails son:

- Ser un framework web de alta productividad compatible con Java.
- Reutilizar tecnologías e implementarlas en un ambiente consistente.
- Ser fácil de aprender y usar.
- Ofrecer documentación suficiente para los desarrolladores.
- Proporcionar un entorno de desarrollo completo.
- Compatibilidad y extensibilidad, principalmente por plugins.

Paradigmas en los cuales se basa:

- Convención sobre configuración: (**CoC**) busca minimizar el número de decisiones que un desarrollador debe de tomar. Dice que en un principio el desarrollador sólo debe de especificar los aspectos no convencionales de la aplicación, y de ahí en adelante el ambiente (o framework) se adaptará a ello y ya no será necesario que el desarrollador haga tareas que ya cubre dicha configuración. Un ejemplo de esto en Grails es que, automáticamente, a los controladores agrega al final del nombre de la clase la palabra “Controller”; y así no es necesario utilizar una notación de Spring para etiquetar a dicha clase como un controlador, y tanto el desarrollador como el framework saben de qué se trata.
- No te repitas: DRY (**Don´t Repeat Yourself**, en inglés) es una filosofía de procesos que dice que no se debe de duplicar ninguna “pieza de información” en un ambiente o sistema. Habla de “pieza de información” de forma amplia, la cual puede referirse a nivel de base de datos, código, plugin, software, documentación o cualquier componente del sistema. La idea de basar el desarrollo en la reutilización no sólo de código sino de cualquier componente del sistema hace que dicho sistema sea altamente consistente, escalable y sobre todo mantenible.

Las características de su entorno de desarrollo son:

- Es orientado a pruebas: (TDD por **Test-Driven Development**, en inglés) es decir que se escriben primero las pruebas y luego el código que haga que las pruebas pasen satisfactoriamente.
- Contiene su propio ORM, llamado GORM el cual está basado en Hibernate.
- Contiene patrones de visualización con GSP (basado en JSP).
- Biblioteca de etiquetas dinámicas.
- Contiene servidor web.
- Tiene recarga automática de recursos.

Grails se ha desarrollado según el paradigma de MVC:

- Modelo: Utiliza GORM (**Grails Object Relational Mapping**), el cual crea las tablas correspondientes en la base de datos según las clases escritas en Groovy (aunque no todas las relaciones entre tablas están soportadas). Contiene métodos dinámicos y estáticos para poder hacer operaciones CRUD. El lenguaje que se utiliza en estos métodos es Hibernate HQL. El mecanismo es implementado mediante Hibernate.
- Controlador: A partir del modelo, Grails puede generar un controlador automáticamente con características generales (para los métodos CRUD). Los controladores dan soporte al comportamiento de las páginas web, y a su vez consumen los métodos que proporciona el modelo. En Grails se separan en dos partes el nivel de controlador: en clases llamadas “Services” y otras llamadas “Controller”. Los Controllers implementan toda la lógica y la funcionalidad de las páginas web a las que dan soporte y cuando necesitan algún tipo de método CRUD ocupan a los Services; los Services contienen métodos CRUD específicos, aquí se plasma la lógica de negocio, y estos ocupan directamente al modelo.
- Vista: Grails soporta a GSJ (**Groovy Server Pages**) y JSP (**Java Server Pages**); las GSP pueden ser generadas automáticamente a partir del Controller; donde por cada método se crea una vista. Estas vistas tienen soportes para una vasta biblioteca de etiquetas propia de Grails, sin embargo puede utilizar otras propias de Java. También es compatible con bibliotecas de Ajax.

Versión utilizada: Grails 2.5.3.

Groovy

Es un lenguaje de programación orientado a objetos (**POO**) implementado sobre la plataforma de Java. Para su creación se basaron en Java, Python, Ruby y Perl. Su estándar es el JSR 241.

Sus principales características son:

- Su sintaxis, modelo de objetos, hilos, procesos y seguridad están basadas en Java.
- El bytecode generado es totalmente compatible con la JVM (**Java Virtual Machine**).
- La mayoría del código Java es válido en Groovy, así que la curva de aprendizaje es muy reducida para un desarrollador Java.
- Tiene un comportamiento dinámico, así que también se usa como lenguaje de scripting.

Versión utilizada: La propia de Grails 2.5.1.

HTML

Lenguaje de marcas de hipertexto (**Hyper Text Markup rLanguaje**, en inglés) es un lenguaje de marcado que se utiliza para hacer páginas web. Es un estándar manejado por W3C y ha sido adoptado por todos los navegadores. La idea básica es que un documento HTML sólo va a contener texto y el navegador será el encargado de interpretarlo;

actualmente siguen saliendo nuevas versiones de HTML por lo tanto los navegadores deben de actualizarse constantemente.

Principales características:

- Se escribe en forma de etiquetas.
- Describe la estructura de un documento, y hasta cierto punto muy limitado su apariencia.
- Puede hacer referencias a scripts y utilizarlos.

Versión utilizada: HTML 5.

IntelliJ IDEA

Es un IDE para desarrollo de software. Tiene dos versiones: comunidad y comercial; la de comunidad es gratis, la comercial no. La de comunidad no tiene soporte para Grails.

Versión utilizada: IntelliJ IDEA 15 (licencia comercial).

JavaScript

Es un lenguaje de script. Su principal uso es en navegadores web, es decir del lado del cliente, aunque también se utiliza del lado del servidor y como complementos para aplicaciones de escritorio. Con excepción del nombre, no tiene nada que ver con el lenguaje Java; tienen semántica diferente y objetivos diferentes. Sus funciones son muy amplias, no sólo para dar funcionalidad e interactividad a la interfaz de la página web por medio de DOM, sino para recibir y mandar información al servidor trabajando junto con Ajax. La gran mayoría de los navegadores actuales soportan JavaScript, y puede ir embebido en el código HTML sin ningún problema.

Sus características son:

- Lenguaje imperativo: Es un paradigma de programación que se basa en un conjunto de instrucciones que dictan cómo realizar una tarea. Así cada paso del algoritmo es una instrucción a hacer.
- Programación basada en prototipos: Es un estilo de POO donde la creación de objetos no es por instanciación sino por clonación de otros objetos o por la escritura de código; así la reutilización de objetos se vuelve muy transparente y sencilla. Se le llama que es un paradigma sin clases.
- Programación orientada a objetos: JavaScript tiene capacidad para funcionar como un lenguaje POO, aunque no puede implementar como tal el concepto de clase, sí puede simular los objetos.

Versión utilizada: JavaScript 5.

MySQL

Ahora MariaDB. Es un Sistema de Gestión de Base de Datos (**DBMS** por sus siglas en inglés) open source de los más populares a nivel mundial, que gestiona una base de datos relacional en lenguaje SQL. Es utilizado comúnmente para desarrollo web. En realidad tiene dos versiones: una pública (GPL o **General Public Licence** en inglés) y otra comercial o privada.

Versión utilizada: Propia de XAMPP 5.5.24.

Plugins

Los plugins que utilicé en el desarrollo de la aplicación son únicamente aquellos que se integran de manera predeterminada a Grails 2.5.3. Los cuales son:

- asset-pipeline-2.5.7
- cache-1.1.8
- database-migration-1.4.0
- famfamfam-1.0.1
- hibernate4-4.3.10
- jquery-1.11.1
- jquery-ui-1.10.4
- scaffolding-2.1.2
- spring-security-core-2.0-RC5
- spring-security-ui-1.0-RC2
- tomcat-7.0.55.3
- webxml-1.4.1

XAMPP

Es un servidor opens source (software libre) que se basa principalmente en la administración de MySQL, Apache e intérpretes de algunos lenguajes como PHP y Perl. Surgió como un acrónimo de:

- **X**: Cualquier sistema operativo.
- **A**: Apache.
- **M**: MySQL (a partir de la versión 5.6.15 es MariaDB).
- **P**: PHP.
- **P**: Perl.

Se le han ido agregando funcionalidades y tecnologías, como OpenSSL y phpMyAdmin.

Versión utilizada: XAMPP 5.5.24.

1.3 Metodología utilizada

MVC

Acrónimo de **Modelo-Vista-Controlador**. Es un patrón de diseño arquitectural que lo que busca es separar la lógica de negocio de la parte de la interfaz de usuario, así como separar los datos de la forma de entrada/salida de estos. La solución que propone es crear tres entidades diferentes: modelo, vista y controlador.

- Modelo: Esta entidad representa el modelo de datos con el cual va a operar el sistema. Aquí es donde se implementa la lógica de negocio, puesto que se encarga de modelar la base de datos, las restricciones y roles que tendrán los usuarios. El modelo sólo se comunica con el controlador.
- Vista: Representa el ambiente o la plataforma adecuada para interactuar con el usuario. Todo lo que el sistema puede hacer es resumido de forma sencilla (es llamada la interfaz de usuario). Es la parte del sistema que el usuario puede ver. La vista sólo se comunica con el controlador.
- Controlador: Es el intermediario entre la Vista y el Modelo. Todas las peticiones que el usuario hace a la Vista el Controlador las atiende; y a su vez este hace las peticiones correspondientes al Modelo para enviar la información requerida a la Vista. Aquí no hay lógica de negocio, sólo la lógica del sistema para funcionar y los algoritmos.

Flujo de control.

1. El usuario interactúa con la Vista.
2. El Controlador recibe la petición de la Vista.
3. El Controlador hace alguna acción CRUD sobre el Modelo.
4. El Controlador pasa la información pertinente a la Vista.
5. La Vista despliega la información al usuario.

En este sistema fue utilizado el patrón MVC dado que el framework Grails está basado en él.

SCRUM

Es una metodología ágil de desarrollo de software el cual utiliza el modelo incremental de desarrollo. A pesar de ser inventada en los años 80´s es hoy en día la metodología ágil más utilizada; básicamente por su sencillez, flexibilidad y alcance. La filosofía de SCRUM es que durante el desarrollo van a surgir cambios de alcances o requerimientos que no pueden ser identificables en un principio; por tanto, no le da peso a planear minuciosamente en un principio. Lo que busca es trabajar de forma que se hagan entregas de trabajo pequeños que se desarrollen en un corto tiempo, con un equipo de trabajo listo para cualquier cambio.

Ciclo de trabajo de SCRUM:
El líder de proyecto es el encargado de marcar desde un principio del proyecto ciertos aspectos muy importantes, como son crear los equipos de trabajo y definir los tiempos de

entrega de avances funcionales. El tiempo de desarrollo se divide en llamados “sprints” que son periodos cortos de tiempo en donde se añade alguna nueva funcionalidad o complejidad al software, pero debe de ser al final del sprint algo utilizable. En cada equipo existe una persona que sabe con claridad los requisitos a alto nivel del proyecto, los cuales se sabe que pueden cambiar a lo largo del desarrollo (que de hecho se esperan que cambien). Se tiene comunicación muy cercana con el cliente.

Las principales características de SCRUM son:

- El modelo de desarrollo es incremental con solapamientos, no en cascada.
- La calidad del producto lo basa en las personas del equipo y el resultado, no en metodologías o practicas empleadas.

Lo que se busca utilizando SCRUM:

- Gestión de las expectativas y deseos del cliente.
- Trabajo anticipado al cambio.
- Modelo de trabajo flexible y fácilmente adaptable.
- Obtener mayor calidad en el software y alta productividad en el desarrollo.
- Reducción de riesgos.
- Maximizar el retorno de inversión.

Reuniones SCRUM: si hay algo que caracteriza a esta metodología son sus reuniones o juntas que representan su naturaleza ágil y pragmática.

- Reunión diaria: al inicio de cada día se hace una junta muy breve que dura de 15 a 20 min, donde cada miembro del equipo involucrado responde a 3 preguntas sencillas: Qué avanzó el día anterior, en qué avanzará a lo largo del día y si existe algún impedimento para que siga avanzando.
- Reunión de sprint: Esta en realidad se divide en dos reuniones: una al inicio y otra al final de cada sprint. La del inicio tiene como objetivo la planificación del sprint; declarar objetivos concretos, crear los equipos de trabajo, marcar reglas específicas, etc. Y al final del sprint se tiene una siguiente reunión para comparar los resultados con lo planeado y obtener retrospectiva y retroalimentación de todo el equipo con el fin de mejorar.

Capítulo 2. Análisis y diseño

Este capítulo está destinado a hablar sobre la manera en que se analizan los requerimientos y el alcance, y con base a eso se diseña el sistema TRACKING.

2.1 Análisis del proceso

El sistema TRACKING está diseñado para agilizar y automatizar ciertos procesos bien definidos, y es por ello que debe de entenderse completamente la problemática para saber qué diseñar.

Primeramente, se debe de tener en cuenta el esquema de mercado y trabajo para el cual está diseñado el sistema, que tiene las siguientes características:

- Productos muy caros que se venden sólo bajo pedido, es decir, cuando el cliente quiere uno debe de hacer el pedido a la empresa.
- Atención especializada al cliente; cada cliente tiene diferentes necesidades y el departamento de ventas y soporte/calidad debe de ir de la mano en el proceso de adquisición: desde que el cliente le define los requerimientos de lo que necesita hasta la configuración y capacitación.
- Dado que el producto es bajo pedido no se tiene inventario; por tanto, la agilidad y coordinación de los departamentos involucrados en el proceso de compra/venta es crucial.

Estas empresas del mismo sector al que está dirigido el sistema comparten un proceso similar para la atención de un nuevo pedido y su seguimiento. El proceso puede resumirse en estos pasos, en su flujo normal:

1. El cliente confirma a la empresa que desea cierto producto.
2. La empresa levanta un pedido hacia su proveedor en el extranjero.
3. El proveedor extranjero recibe el pedido y comunica a la empresa que el producto va de salida.
4. El producto llega a la empresa.
5. El producto debe ser inspeccionado para asegurar que esté libre de fallos.
6. El producto se le envía al cliente.
7. El cliente recibe el producto.

A partir de este proceso se puede definir el ciclo de vida de un pedido:

1. Nuevo pedido generada.
2. Pedido solicitado a proveedor.
3. Pedido atendido por proveedor.
4. Pedido enviado por el proveedor.
5. Pedido recibido en la empresa.
6. Pedido inspeccionado.
7. Pedido enviado al cliente.
8. Pedido recibido por el cliente.

Los departamentos que intervienen a lo largo del ciclo de vida de un pedido son:

- Ventas: es el primer y continuo contacto que tiene el cliente con la empresa y debe de darle continua retroalimentación del estatus de su pedido.

- Compras: es el encargado de hacer los pedidos al proveedor. Es el contacto con los proveedores.
- Logística: se encarga de recibir y enviar el pedido al cliente.
- Soporte técnico / calidad: se encarga de hacer la inspección del producto una vez que llegue a la empresa. Además de dar asesoría técnica y configurar el equipo en caso de ser requerido.

Bajo este esquema de trabajo las empresas necesitan saber el estado del pedido en todo momento además de mostrar el panorama completo del proceso. Con todo esto se busca tener el máximo control sobre cada pedido. Logrando este control se obtiene:

- El mínimo tiempo de entrega al cliente.
- Detectar problemas en el ciclo de vida del producto y solucionarlos inmediatamente.
- Evitar pérdidas de dinero y tiempo por no aplicar bien el proceso.

Por lo tanto, las necesidades que se buscan cubrir con el sistema TRACKING son:

1. Excelente atención al cliente: Excelencia de calidad en el servicio de atención al cliente. Los clientes, que están pagando una buena cantidad de dinero por el producto, esperan un servicio personalizado y eficaz.

2. Control de logística de pedidos: La eficacia de la logística es crucial para obtener rentabilidad en la venta de productos por pedido de importación; dado que intervienen varios departamentos y existe una gran dependencia entre ellos se deben evitar a toda costa retrasos de tempo incensarios, pues eso equivale a grandes pérdidas de dinero.

3. Automatización de procesos: Las actividades y comunicación entre el departamento de compras, ventas, logística y soporte técnico/calidad deben de estar perfectamente bien sincronizados y la información que todos utilizan debe estar centralizada y bien controlada.

Por lo tanto, un pedido será el objeto principal del sistema TRACKING, el cual tendrá un ciclo de vida representado por las siguientes etapas:

1. Orden de compra: se le solicita al proveedor.
2. Embarque: se envía por el proveedor.
3. Arribo: recibido por la empresa.
4. Revisión: inspeccionado por la empresa.
5. Envío: enviado al cliente.
6. Recepción: recibido por el cliente.

Y el estatus de un pedido será sólo “Abierto” y “Cerrado”. Cuando se crea un nuevo pedido no tiene ninguna etapa asociada y tiene estatus “Abierto”. Debe de ir pasando de etapa a etapa en el orden ya descrito. Cuando llegue a la última etapa su estatus cambia a “Cerrado”.

2.2 Análisis de requerimientos

A partir del análisis del proceso visto desde varios puntos de vista ahora sigue el análisis de los requerimientos; a partir de los cuales se divide el sistema en módulos (Tabla 1) que se encargan de satisfacer a uno o varios requerimientos.

Módulo	Requerimientos
Login	Permitir el acceso del usuario con su username y contraseña.
Vendedores	Crear, editar, ver y listar vendedores.
Clientes	Crear, editar, ver y listar clientes.
Proveedores	Crear, editar, ver y listar proveedores.
Líneas de Negocio	Crear, editar, ver y listar líneas de negocio (áreas del departamento de venta).
Pedidos	Crear, editar, ver y listar pedidos.
Tiempos	Crear, editar, ver y listar los tiempos por etapa que cada proveedor tiene estimado para atender un pedido de producto.
Administración	• Administrar la evolución de cada pedido por sus etapas del ciclo de vida. • Ver la relación de los pedidos por cada etapa del ciclo de vida.
Estadísticas	• Generar gráficas de relación de pedidos por mes. • Generar tabla de movimientos recientes del sistema. • Generar gráfica de porcentaje de ventas de las líneas de negocio. • Generar gráfica de relación de pedidos por cliente.

Tabla 1. Módulos del sistema.

Los roles que van a existir para los usuarios son los siguientes (Tabla 2).

Rol	Descripción
Admin	Se encarga de la configuración del sistema. Tiene acceso a todos los módulos.
User	Se encarga del uso corriente del sistema, usando sólo la configuración establecida por el administrador y no puede cambiarla. Tiene acceso a algunos módulos.

Tabla 2. Roles del sistema.

El acceso que los roles tienen a los distintos módulos es el siguiente (Tabla 3).

Módulo	Admin	User
Login	Sí	Sí
Vendedores	Sí	No
Clientes	Sí	No
Proveedores	Sí	No
Líneas de Negocio	Sí	No
Pedidos	Sí	No
Tiempos	Sí	No
Administración	Sí	Sí
Estadísticas	Sí	Sí

Tabla 3. Acceso a módulos del sistema por roles.

2.3 Casos de uso

A continuación, se describen los distintos escenarios de uso que puede tener el sistema (Tabla 4, Tabla 5, Tabla 6, Tabla 7, Tabla 8, Tabla 9, Tabla 10, Tabla 11, Tabla 12).

Módulo: Administración				
Caso de uso	**Pre-condición**	**Pos-condición**	**Flujo normal**	**Excepciones**
Orden de compra	Debe existir el número de pedido y no haber sido procesado aún para dicha etapa.	El pedido es asociado con una orden de compra y queda en dicha etapa.	1. Ingresar el número de pedido a promover y la orden de compra a asociar (puede ser una existente o una nueva). 2. Escribir la fecha y una descripción breve. 3. Su estatus se actualiza a “Orden de compra”. 4. Guardar etapa.	1. El número de pedido no existe o ya está en dicha etapa.
Embarque	Debe existir el número de pedido, no haber sido procesado aún para dicha etapa, estar justo en la etapa anterior e ingresar una fecha mayor o igual a la fecha de la etapa anterior.	El pedido se actualiza a dicha etapa.	1. Ingresa el número de pedido. 2. Escribe la fecha y descripción breve. 3. Su estatus se actualiza a “Embarque”. 4. Guarda etapa.	1. El número de pedido no existe o ya está en dicha etapa o no está justo en la etapa anterior o la fecha es menor a la fecha anterior.
Arribo	Debe existir el número de pedido, no haber sido procesado aún para dicha etapa, estar justo en la etapa anterior e ingresar una fecha mayor o igual a la fecha de la etapa anterior.	El pedido se actualiza a dicha etapa.	1. Ingresa el número de pedido. 2. Escribe la fecha y descripción breve. 3. Su estatus se actualiza a “Arribo”. 4. Guarda etapa.	1. El número de pedido no existe o ya está en dicha etapa o no está justo en la etapa anterior o la fecha es menor a la fecha anterior.
Revisión	Debe existir el número de pedido, no haber sido procesado aún para dicha etapa, estar justo en la etapa anterior e ingresar una fecha mayor o igual a la fecha de la etapa anterior.	El pedido se actualiza a dicha etapa.	1. Ingresa el número de pedido. 2. Escribe la fecha y descripción breve. 3. Su estatus se actualiza a “Revisión”. 4. Guarda etapa.	1. El número de pedido no existe o ya está en dicha etapa o no está justo en la etapa anterior o la fecha es menor a la fecha anterior.

Envío	Debe existir el número de pedido, no haber sido procesado aún para dicha etapa, estar justo en la etapa anterior e ingresar una fecha mayor o igual a la fecha de la etapa anterior.	El pedido se actualiza a dicha etapa.	1. Ingresa el número de pedido. 2. Escribe la fecha y descripción breve. 3. Agregar nombre de paquetería y número de guía. 4. Su estatus se actualiza a “Envío”. 5. Guarda etapa.	1. El número de pedido no existe o ya está en dicha etapa o no está justo en la etapa anterior o la fecha es menor a la fecha anterior.
Recepción	Debe existir el número de pedido, no haber sido procesado aún para dicha etapa, estar justo en la etapa anterior e ingresar una fecha mayor o igual a la fecha de la etapa anterior.	El pedido se actualiza a dicha etapa.	1. Ingresa el número de pedido. 2. Escribe la fecha y descripción breve. 3. Su estatus se actualiza a “Cerrado”. 4. Guarda etapa.	1. El número de pedido no existe o ya está en dicha etapa o no está justo en la etapa anterior o la fecha es menor a la fecha anterior.
Consultar pedidos por etapa	N/A	Se verá una lista de todos los pedidos que están en la misma etapa	1. Ir a Home. 2. Ir a la sección “Total de pedidos”. 3. Elegir la etapa de la cual se quieran ver todos los pedidos que están en ella. 4. Seleccionar la etapa.	N/A

Tabla 4. Casos de uso de módulo Administración.

Módulo: Estadísticas				
Caso de uso	**Pre-condición**	**Pos-condición**	**Flujo normal**	**Excepciones**
Gráfica de pedidos por mes	Debe de haber por lo menos un pedido en lo que va del año.	Ver gráfica que muestras todos los pedidos de cada mes durante el año en curso.	1. Ir a Home. 2. Ir al área de “Gráfica de pedidos”. 3. La gráfica se genera automáticamente.	1. No ha habido ningún pedido en el año y no muestra nada la gráfica.
Tabla de movimientos recientes	N/A	Ver tabla con las 10 últimas actualizaciones de estatus, viendo las columnas: Etapa, Número de pedido, Orden de compra, Fecha de cambio.	1. Ir a Home. 2. Ir al área de “Movimientos recientes”. 3. La tabla se genera automáticamente.	N/A

Pedidos por etapa	N/A	Ver una tabla con todos los pedidos que están en una misma etapa; con las columnas: Número de pedido, Órden, Vendedor, Cliente, Status, Archivo.	1. Ir a Home. 2. Ir al parea de “Total de pedidos”. 3. Elegir la etapa que se quiera visualizar. 4. Se abrirá una nueva ventana con una tabla de todos los pedidos que se encuentran en la etapa seleccionada.	N/A
Gráfica de ventas	Debe existir al menos una línea de negocio y un pedido en lo que va del año.	Ver gráfica que muestra cada línea de negocio con la cantidad de pedidos asociados a ellas en lo que va del año.	1. Ir a Home. 2. Ir al área de “Ventas por departamento”. 3. La gráfica se genera automáticamente.	1. No existe ningún departamento. 2. No existe ningún pedido en lo que va del año.
Gráfica mejores clientes	N/A	Ver gráfica de pastel que muestra el porcentaje de pedidos que ha hecho los 10 mejores clientes en lo que va del año. Ver tabla que muestra el total de pedidos de los 10 mejores clientes en lo que va del año.	1. Ir a Home. 2. Ir al área de “Pedidos por cliente”. 3. La gráfica se genera automáticamente.	N/A
Pedidos por cliente	Debe de haber por lo menos un cliente.	Ver una tabla con todos los pedidos que están que ha hecho cierto cliente; con las columnas: Número de pedido, Órden, Vendedor, Cliente, Status, Archivo.	1. Ir a Home. 2. Ir al área de “Pedidos por cliente”. 3. En la “Tabla de clientes y sus pedidos” selecciona el nombre del cliente del cual se quiere ver los pedidos. 4. Se abrirá una nueva ventana con una tabla de todos los pedidos que del cliente seleccionado.	N/A

Tabla 5. Casos de uso de módulo Estadísticas.

Módulo: Login				
Caso de uso	**Pre-condición**	**Pos-condición**	**Flujo normal**	**Excepciones**
Login	Tener datos válidos en BD.	Entrar al sistema.	1. Escribir usuario y contraseña. 2. Entrar al sistema.	1. Usuario o contraseña incorrectos y se niega el acceso.

Tabla 6. Casos de uso de módulo Login.

Módulo: Vendedores				
Caso de uso	**Pre-condición**	**Pos-condición**	**Flujo normal**	**Excepciones**
Crea vende-dor	N/A	Nuevo vendedor.	1. Escribir nombre y email. 2. Crear.	1. Escribir email en formato erróneo. 2. Escribir email que ya existe.
Lista vende-dores	N/A	Ver todos los vendedo-res.	1. Ver vendedores.	N/A
Muestra vende-dor	Debe existir el vendedor.	Ver detalles de vendedor.	1. Muestra detalles del vendedor.	N/A
Edita vende-dor	Debe existir el vendedor.	Vendedor editado.	1. Editar los campos. 2. Guardar cambios.	1. Escribir email en formato erróneo. 2. Escribir email que ya existe.
Elimina vende-dor	Debe existir el vendedor.	Vendedor eliminado.	1. Confirmar eliminación 2. Eliminar	N/A

Tabla 7. Casos de uso de módulo Vendedores.

Módulo: Clientes				
Caso de uso	**Pre-condición**	**Pos-condición**	**Flujo normal**	**Excepciones**
Crea cliente	N/A	Nuevo cliente.	1. Escribir nombre y email. 2. Crear.	1.Escribir email en formato erróneo. 2.Escribir email que ya existe.
Lista clientes	N/A	Ver todos los clientes	1. Ver clientes.	N/A
Muestra cliente	Debe existir el cliente.	Ver detalles de cliente.	1. Muestra detalles del cliente.	N/A
Edita cliente	Debe existir el cliente.	Cliente editado.	1. Editar los campos. 2. Guardar cambios.	Escribir email en formato erróneo. Escribir email que ya existe
Elimina cliente	Debe existir el cliente.	Cliente eliminado.	1. Confirmar eliminación. 2. Eliminar.	N/A

Tabla 8. Casos de uso de módulo Clientes.

Módulo: Proveedores				
Caso de uso	**Pre-condición**	**Pos-condición**	**Flujo normal**	**Excepciones**
Crea proveedor	N/A	Nuevo proveedor.	1. Escribir nombre. 2. Crear.	N/A
Lista proveedores	N/A	Ver todos los proveedores.	1. Ver proveedores.	N/A
Muestra proveedor	Debe existir el proveedor.	Ver detalles de proveedor.	1. Muestra detalles del proveedor.	N/A
Edita proveedor	Debe existir el proveedor.	Proveedor editado.	1. Editar los campos. 2. Guardar cambios.	N/A
Elimina proveedor	Debe existir el proveedor	Proveedor eliminado.	1. Confirmar eliminación. 2. Eliminar.	N/A

Tabla 9. Casos de uso de módulo Proveedores.

Módulo: Línea de Negocio				
Caso de uso	**Pre-condición**	**Pos-condición**	**Flujo normal**	**Excepciones**
Crea línea	N/A	Nueva línea.	1. Escribir nombre. 2. Crear.	N/A
Lista líneas	N/A	Ver todos las líneas	1. Ver líneas.	N/A
Muestra línea	Debe existir la línea.	Ver detalles de línea.	1. Muestra detalles de línea.	N/A
Edita línea	Debe existir la línea.	Línea editada.	1. Editar los campos. 2. Guardar cambios.	N/A
Elimina línea	Debe existir la línea.	Línea eliminada.	1. Confirmar eliminación. 2. Eliminar.	N/A

Tabla 10. Casos de uso de módulo Línea de Negocio.

Módulo: Pedidos				
Caso de uso	**Pre-condición**	**Pos-condición**	**Flujo normal**	**Excepciones**
Crea pedido	N/A	Nuevo pedido, por defecto su estatus es “Abierto”.	1. Escribir número de pedido, fecha de creación; seleccionar vendedor, área (línea de negocio), cliente, proveedor; adjuntar cotización. 2. Crear.	1.Escribir un número de pedido que ya existe.
Lista pedidos	N/A	Ver pedidos	1. Ver pedidos.	N/A
Muestra pedido	Debe existir el pedido.	Ver detalles de pedido.	1. Muestra detalles del pedido.	N/A

Edita pedido	Debe existir el pedido.	Pedido editado.	1. Editar los campos. 2. Guardar cambios.	1. Escribir un número de pedido que ya existe.
Resumen de pedido	Debe existir el pedido	Ver resumen de pedido.	1. Escribir el número de pedido. 2. Ver resumen.	N/A

Tabla 11. Casos de uso de módulo Pedidos.

Módulo: Tiempos				
Caso de uso	**Pre-condición**	**Pos-condición**	**Flujo normal**	**Excepciones**
Asigna tiempos	Debe existir al menos un proveedor.	Nueva asignación de tiempos.	1. Escribir el número de días estimados que va a tomar cubrir cada etapa del pedido. 2. Crear.	N/A
Lista de proveedores con tiempos	N/A	Ver lista de proveedores con tiempos asignados y el total.	1. Ver tiempos.	N/A
Muestra tiempos de proveedor	Debe existir la asignación.	Ver detalles de tiempos.	1. Muestra detalles de los tiempos.	N/A
Edita tiempos	Debe existir la asignación.	Tiempos editados.	1. Editar los tiempos. 2. Guardar cambios.	N/A

Tabla 12. Casos de uso de módulo Tiempos.

2.4 Diagrama Entidad-Relación

La base de datos del sistema TRACKING se compone de la siguiente manera (Figura 1).

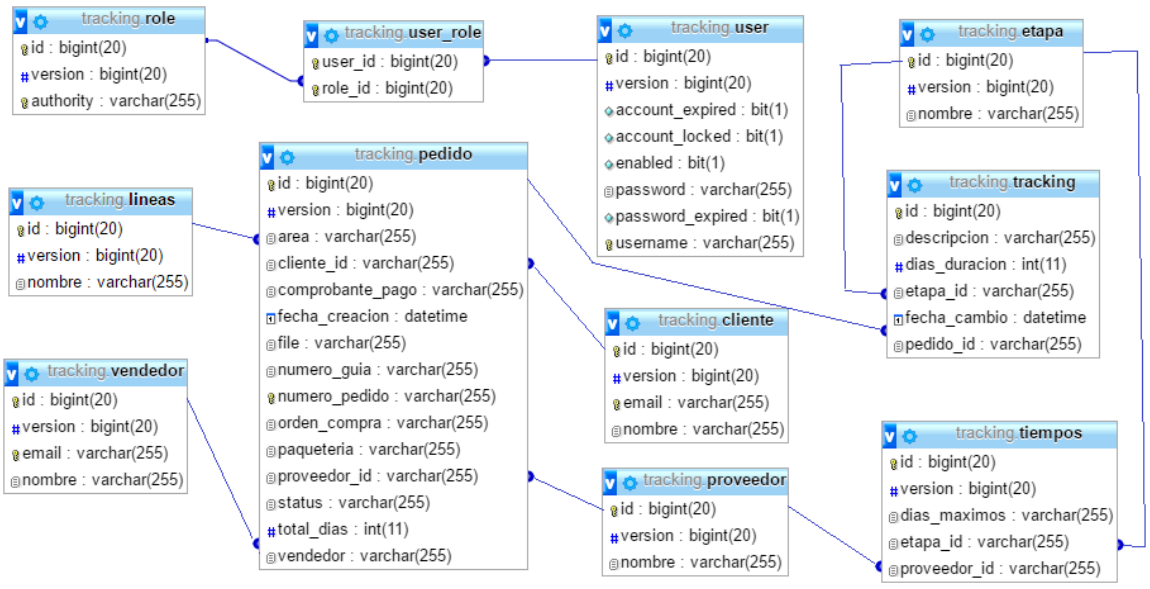


Figura 1. Diagrama Entidad-Relación del modelo de datos del sistema.

2.5 Diccionario de datos

A continuación, se describen las entidades de la base de datos con sus respectivos atributos (Tabla 13, Tabla 14, Tabla 15, Tabla 16, Tabla 17, Tabla 18, Tabla 19, Tabla 20, Tabla 21, Tabla 22, Tabla 23).

Entidad: Cliente				
Descripción: Clientes que adquieren los productos de la empresa.				
Atributo	**Tipo**	**¿Permite nulos?**	**¿Único?**	**Descripción**
id	Bigint(20)	No	Sí	Identificador único en la BD.
email	Varchar(255)	No	Sí	Correo electrónico del cliente, debe ser único.
nombre	Varchar(255)	No	No	Nombre del cliente.

Tabla 13. Datos de la entidad Cliente.

Entidad: Etapa				
Descripción: Catálogo con las etapas en las que puede estar un pedido a lo largo de su ciclo de vida.				
Atributo	**Tipo**	**¿Permite nulos?**	**¿Único?**	**Descripción**
id	Bigint(20)	No	Sí	Identificador único en la BD.
nombre	Varchar(255)	No	No	Nombre de la etapa.

Tabla 14. Datos de la entidad Etapa.

Entidad: Pedido				
Descripción: Pedido de un producto que el cliente solicita a la empresa; la cual tendrá asociado otras entidades como son: un cliente, un departamento de venta, un vendedor y un proveedor.				
Atributo	**Tipo**	**¿Permite nulos?**	**¿Único?**	**Descripción**
id	Bigint(20)	No	Sí	Identificador único en la BD.
area	Varchar(255)	No	No	Id de Línea asociada (departamento de ventas).
cliente_id	Varchar(255)	No	No	Id de Cliente asociado.
fecha_creacion	datetime	Sí	No	Fecha en que se genera.
file	Varchar(255)	Sí	No	Dirección del archivo adjunto.
numero_guia	Varchar(255)	Sí	No	Número de guía de la paquetería.
numero_pedido	Varchar(255)	No	Sí	Correo electrónico del cliente, debe ser único.
orden_compra	Varchar(255)	Sí	No	Nombre de la orden de compra a la que se le asigna.
paqueteria	Varchar(255)	Sí	No	Nombre de la paquetería por la cual se le manda al cliente.
proovedor_id	Varchar(255)	No	No	Id del Proveedor asociado.
status	Varchar(255)	Sí	No	En qué etapa se encuentra el Pedido.
Total_dias	int(11)	No	No	Días estimados que va a tardar en entregarse el pedido según el proveedor asociado.
vendedor	Varchar(255)	No	No	Id del Vendedor asignado.

Tabla 15. Datos de la entidad Pedido.

Entidad: Lineas				
Descripción: Líneas de negocio o departamentos de ventas de la empresa.				
Atributo	**Tipo**	**¿Permite nulos?**	**¿Único?**	**Descripción**
id	Bigint(20)	No	Sí	Identificador único en la BD.
nombre	Varchar(255)	No	No	Nombre del departamento.

Tabla 16. Datos de la entidad Lineas.

Entidad: Proveedor				
Descripción: Proveedores de la empresa.				
Atributo	**Tipo**	**¿Permite nulos?**	**¿Único?**	**Descripción**
id	Bigint(20)	No	Sí	Identificador único en la BD.
nombre	Varchar(255)	No	No	Nombre del proveedor.

Tabla 17. Datos de la entidad Proveedor.

Entidad: Rol				
Descripción: Diferentes roles que habrá para los usuarios del sistema				
Atributo	**Tipo**	**¿Permite nulos?**	**¿Único?**	**Descripción**
id	Bigint(20)	No	Sí	Identificador único en la BD.
authority	Varchar(255)	No	No	Nombre del rol.

Tabla 18. Datos de la entidad Rol.

Entidad: Tiempos				
Descripción: Días máximos estimados en que un pedido se va a tardar en llegar a cierta etapa según el proveedor del producto.				
Atributo	**Tipo**	**¿Permite nulos?**	**¿Único?**	**Descripción**
id	Bigint(20)	No	Sí	Identificador único en la BD.
dias_maximo	Varchar(255)	No	No	Días estimados.
etapa_id	Varchar(255)	No	No	Id de la etapa.
proveedor_id	Varchar(255)	No	No	Id del proveedor.

Tabla 19. Datos de la entidad Tiempos.

Entidad: Tracking				
Descripción: Gestión del ciclo de vida de los pedidos, donde cada pedido se va relacionando con una nueva etapa y se contabiliza cuánto tiempo va tardando en pasar a de una etapa a otra.				
Atributo	**Tipo**	**¿Permite nulos?**	**¿Único?**	**Descripción**
id	Bigint(20)	No	Sí	Identificador único en la BD.
dias_duracion	int(11)	No	No	Días que tardó en pasar a dicha etapa.
etapa_id	Varchar(255)	No	No	Id de la etapa a la que pasó.
fecha_cambio	datetime	Sí	No	Fecha en que se hizo la promoción de etapa.
pedido_id	Varchar(255)	No	No	Id del pedido que se está manejando.

Tabla 20. Datos de la entidad Tracking.

Entidad: User				
Descripción: Usuario que utiliza el sistema.				
Atributo	**Tipo**	**¿Permite nulos?**	**¿Único?**	**Descripción**
id	Bigint(20)	No	Sí	Identificador único en la BD.
account_expired	Bit(1)	No	No	¿Cuenta expirada?
account_locked	Bit(1)	No	No	¿Cuenta bloqueada?
enabled	Bit(1)	No	No	¿Cuenta activada?
password	Varchar(255)	No	No	Contraseña encriptada.
password_expired	Bit(1)	No	No	¿Contraseña expirada?
username	Varchar(255)	No	Sí	Nombre de usuario.

Tabla 21. Datos de la entidad User.

Entidad: User_role				
Descripción: Relaciona los usuarios con los roles que puede tener.				
Atributo	**Tipo**	**¿Permite nulos?**	**¿Único?**	**Descripción**
user_id	Bigint(20)	No	No	Identificador del usuario
role_id	Bigint(20)	No	No	Identificador del rol

Tabla 22. Datos de la entidad User_role.

Entidad: Vendedor				
Descripción: Vendedores de la empresa.				
Atributo	**Tipo**	**¿Permite nulos?**	**¿Único?**	**Descripción**
id	Bigint(20)	No	Sí	Identificador único en la BD.
email	Varchar(255)	No	Sí	Correo electrónico del vendedor, debe ser único.
nombre	Varchar(255)	No	No	Nombre del vendedor.

Tabla 23. Datos de la entidad Vendedor.

Capítulo 3. Desarrollo

En este capítulo se verán los pasos exactos y métodos con los cuales se llevó a cabo el desarrollo del sistema: configuración específica, código y flujo del sistema.

El nombre del proyecto en Grails se llama *Tracking*. La estructura del proyecto Tracking viene dada de la siguiente manera (Figura 2).

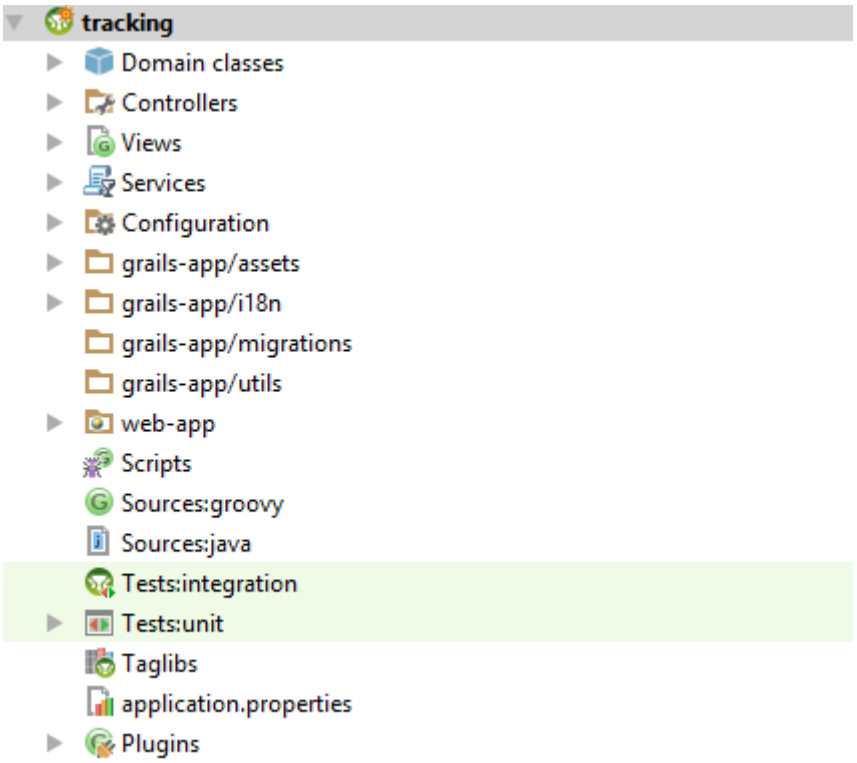


Figura 2. Estructura general del proyecto.

La carpeta de clases de Dominio (Domain classes) contiene todos los dominios que se definen, es decir, las clases propias del ORM del Grails que mapean a la base de datos (Figura 3).

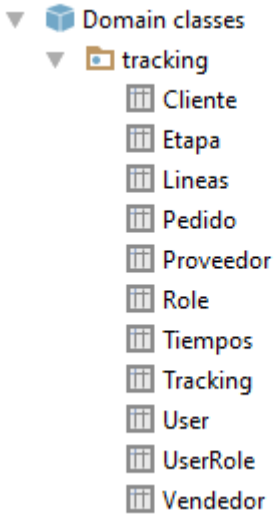


Figura 3. Dominios del sistema.

La carpeta Controladores (Controllers) contiene todos los controladores de la aplicación (Figura 4).

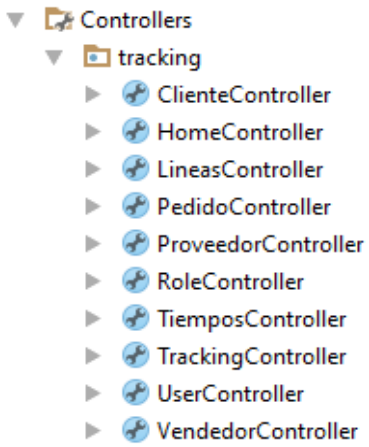


Figura 4. Controladores del sistema.

La carpeta Servicios (Services) contiene todos los servicios de la aplicación (Figura 5).

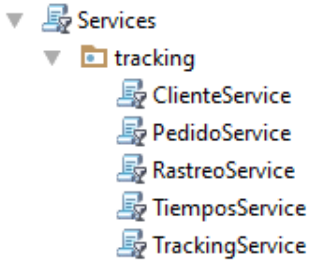


Figura 5. Servicios del sistema.

La carpeta Vistas (Views) contiene todas las vistas de la aplicación separadas por carpetas que por lo regular corresponden a un controlador. Y cada vista (GSP) corresponde a un método del controlador correspondiente (Figura 6).

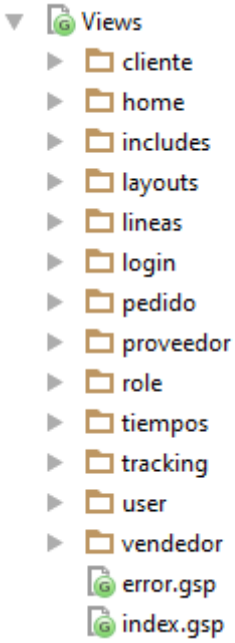


Figura 6. Vistas del sistema.

La carpeta de configuración viene dada por los siguientes archivos (Figura 7).

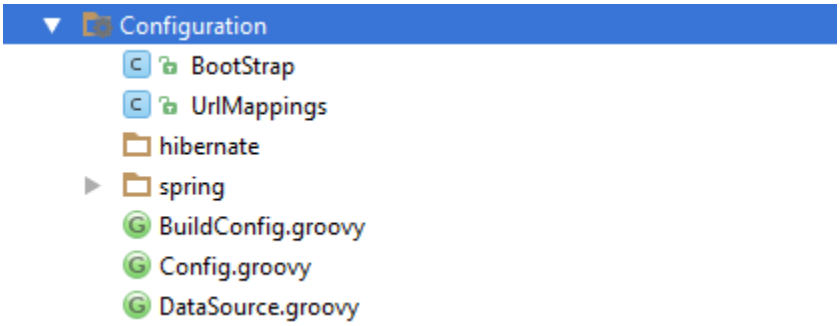


Figura 7. Configuración del sistema.

Las demás carpetas se mencionarán si es necesario en los puntos específicos de cada módulo.

3.1 Archivos de módulo

La relación entre los módulos y sus archivos correspondientes, a nivel de MVC, viene dada por la Tabla 24.

Módulo	Dominio	Controlador	Servicio	Vistas
Login	Role User UserRole	RoleController UserController	N/A	Layouts/Spring-SecurityUI login/auth role/create|edit|search user/create|edit|search
Vendedores	Vendedor	VendedorController	N/A	vendedor/_form|create|edit|index|show
Clientes	Cliente	ClienteController	N/A	cliente/_form|créate|edit|index|show
Proveedores	Proveedor	ProveedorController	N/A	proveedor/_form|create|edit|index|show
Líneas de Negocio	Lineas	LineasController	N/A	lineas/_form|create|edit|index|show
Pedidos	Pedido	PedidoController	PedidoService TrackingService	pedido/_form|create|edit|index|show
Tiempos	Tiempos	TiemposController	TiemposService	tiempos/_form|créate|edit|index|show
Administración	Etapa Tracking	TrackingController	TrackingService	tracking/arribo|embarque|envio|ordenDeCompra|recepcion tracking/revision
Estadísticas	N/A	PedidoController HomeController	ClienteService RatreoService TrackingService	pedido/_pedido|resumen home/index|pedidosClientes|pedidosEtapa

Tabla 24. Dominos, controladores, servicios y vistas de cáda módulo del sistema.

3.2 Conexión con la base de datos y servidor

Para que los datos de la aplicación tengan persistencia se configuró la forma de conectarse con MySQL (ahora MariaDB). Primero se instala XAMPP 5.5.24 (https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/5.5.24/).

Graill debe configurarse para que se conecte con MySQL dentro del archivo *DataSource* en la sección de *development*:

```
dataSource {
    driverClassName="com.mysql.jdbc.Driver"
    dbCreate = "update"
    url = "jdbc:mysql://localhost:3306/tracking"
    username="root"
    password=""
}
```

Donde el nombre de la base de datos que se va a ocupar es “tracking” y el username y password que se agregan son los establecidos de manera predeterminada en MySQL. Luego se accede a phpMyAdmin (en http://localhost/xampp/index.php seleccionar *phpMyAdmin*; o desde XAMPP seleccionar *Admin* de MySQL) y se crea una nueva base de datos vacía (Figura 8).

Figura 8. Creación de la base de datos.

Luego se descarga un conector entre MySQL y java (JVM) que es mysql-connector-java-5.1.18-bin (https://dev.mysql.com/downloads/connector/j/5.1.html). El cual se debe de meter dentro de la carpeta *lib* del proyecto.

El plugin para conectar Grails con Tomcat ya se encuentra en la configuración en *BuildConfig* en el procedure *plugins*:

```
// plugins for the build system only
build ":tomcat:7.0.55.3" // or ":tomcat:8.0.22"
```

Con todo lo anterior la aplicación ya está lista para usarse como aplicación web que persiste datos. Lo único que se debe de hacer cada vez que se quiera utilizar el sistema TRACKING es ejecutar XAMPP junto con Apache, MySQL y Tomcat (Figura 9).

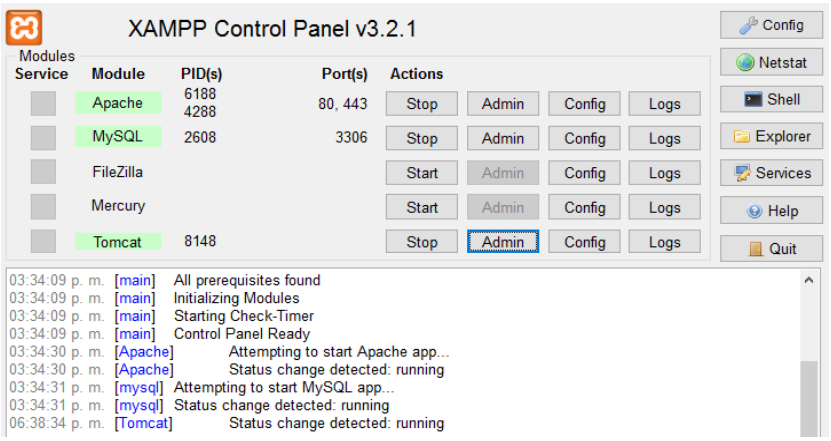


Figura 9. Panel de control de XAMPP.

3.3 Ambiente gráfico

Grails ya contiene un ambiente gráfico sencillo con diseño propio que soporta tecnologías como Bootstrap, jQuery, Ajax, Angular, etc (Figura 10).

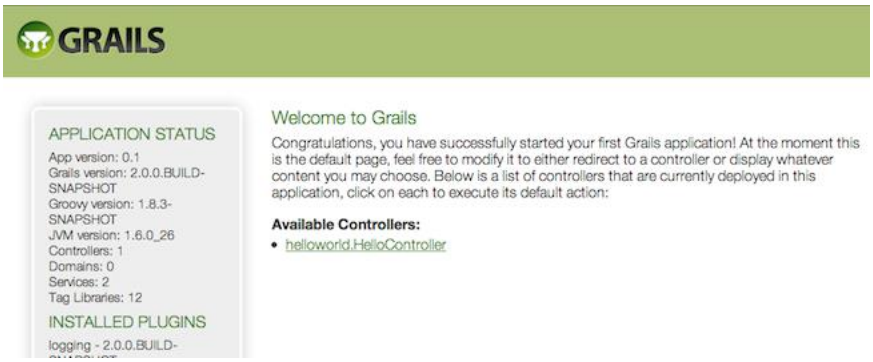


Figura 10. Pantalla virgen de la aplicación web.

El diseño predeterminado que tiene Grails ya está listo para usarse, sin embargo, es sencillo y tiene muchas limitaciones. En su lugar decidí utilizar un ambiente completamente nuevo y mucho más completo. Principalmente por la necesidad de utilizar gráficas e interacciones complejas con el usuario.

Material Admin Responsive es un conjunto de plantillas inspiradas en el diseño de Google que contiene métodos jQuery y Angular para implementar interacciones complejas con el usuario de manera gráfica. Se puede descargar desde la página https://wrapbootstrap.com/theme/material-admin-responsive-angularjs-WB011H985.

La carpeta donde se encuentran todos los archivos se llama *materialadmin* la cual la agregué a la carpeta *web-app* del proyecto; que es donde se encuentran todos los recursos de los módulos que se irán viendo más adelante. Lo único que se debe de hacer para poder utilizar las funciones predeterminadas de *materialadmin* es importar en el código los archivos jquery que se quieran utilizar. (No existe como tal documentación de cómo utilizar el *materialadmin*, más bien se basa en ejemplos prácticos los cuales se deben de adaptar a lo que requiera.)

Por ejemplo, en la vista *home* se tiene la gráfica *Top Mejores Clientes* que es una gráfica de pastel que relaciona a los clientes con el número de pedidos que han hecho en el año (Figura 11).

Figura 11. Gráfica *Top Mejores Clientes*.

Para poder utilizar la gráfica de pastel únicamente se debe de hacer referencia al archivo JS que contiene las funciones y luego agregar dentro de las etiquetas script el código con la funcionalidad que se requiere para la gráfica.

```
<script
      src="${resource(dir:'materialadmin/vendors/bower_components/flot',file:'jquery
      .flot.pie.js')}"></script>

<script>
          $(document).ready(function(){
              var pieData= new Array();
              <g:each in="${topClientes}" var="top">
              pieData.push({
                  data: ${top.total},
                  label: '${top.cliente}',
              });
              </g:each>
              /* Pie Chart */
              if($('#pie-chart-Top-Clientes')[0]){
                  $.plot('#pie-chart-Top-Clientes', pieData, {
                      series: {
                          pie: {
                              show: true,
                              stroke: {
                                  width: 2,
                              },
                          },
                      },
                      legend: {
                          container: '.flc-pie',
                          backgroundOpacity: 0.5,
                          noColumns: 0,
                          backgroundColor: "white",
                          lineWidth: 0
                      },
                      grid: {
                          hoverable: true,
                          clickable: true
                      },
                      tooltip: true,
                      tooltipOpts: {
                          content: "%p.0%, %s", // show percentages, rounding to 2
      decimal places
                          shifts: {
                              x: 20,
                              y: 0
                          },
                          defaultTheme: false,
                          cssClass: 'flot-tooltip'
                      }
                  });
              }
          });
      </script>
```

Lo único que se está haciendo en el script es pasar los parámetros a la función *pieData* para que se comporte de la manera que se requiere.

3.4 Login

Instalación y configuraciones generales del plugin.

Para hacer el login de la aplicación utilicé el plugin Spring-Security 2.0 (http://grails.org/plugin/spring-security-core). En la url del plugin viene documentación más específica de cómo utilizarse, así que aquí pondré lo más básico.
Lo primero es agregarlo al archivo *BuildConfig* en el procedure *plugins*:

```
plugins {
    compile ":spring-security-core:2.0-RC5"
}
```

Así la próxima vez que se compile la aplicación se descarga e instala el plugin, pero sólo la primera vez. Siguiendo el manual de instalación se deben generar los siguientes Dominios, Controladores y Vistas:

- Role
- User
- UserRole
- RoleController
- UserController
- UserRoleController
- auth.gsp

Después se instala el plugin Spring-Security-UI 1.0 (https://grails.org/plugin/spring-security-ui) el cual agrega una interfaz gráfica y varias funcionalidades, que a pesar de que no se ocuparán en Tracking se genera escalabilidad a la aplicación.

```
plugins {
    compile ":spring-security-ui:1.0-RC2"
}
```

Así la próxima vez que se compile la aplicación se descarga e instala el plugin, pero sólo la primera vez. Siguiendo el manual de instalación se debe generar la Vista:

- springSecurityUI.gsp

Se deben de agregar los permisos generales que tiene Spring Security en el archivo *Config*:

```
grails.plugin.springsecurity.userLookup.userDomainClassName = 'tracking.User'
grails.plugin.springsecurity.userLookup.authorityJoinClassName =
'tracking.UserRole'
grails.plugin.springsecurity.authority.className = 'tracking.Role'
```

Ahora para crear usuarios y contraseñas se abre *BootStrap*, el cual es un archivo que Grails ejecuta al iniciar la aplicación. Se agrega dentro del procedure *init*:

```
def init = { servletContext ->
    def adminRole = new Role(authority: 'ROLE_ADMIN').save(flush: true)
    def userRole = new Role(authority: 'ROLE_USER').save(flush: true)
    def testUser = new User(username: 'admin', password: 'admin')
    testUser.save(flush: true)
    def testUser2 = new User(username: 'user', password: 'user')
    testUser2.save(flush: true)
    UserRole.create testUser, adminRole, true
    UserRole.create testUser2, userRole, true
    assert User.count() == 2
    assert Role.count() == 2
    assert UserRole.count() == 2
}
```

Con lo cual se crea el rol Admin y el rol User; se crea un usuario con username “admin” y password “admin” al cual se le asigna el rol Admin y otro con username “user” y password “user” al cual se le asigna el rol User. Este código se ejecuta una sola vez, pero debe de estar en dicha clase por si se desea a borrar o reiniciar la base de datos.

Configurar los permisos de la aplicación.

Spring Security se basa en usuarios y roles, donde la relación es N:N. La forma de controlar el acceso de los usuarios a los módulos a la aplicación a nivel controlador es por medio de la anotación *@Secured*, a la cual se le agrega una lista con los roles o usuarios que tienen el permiso de acceso a cierta clase o cierto método.

Ejemplo: *TiempoController* pertenece a un módulo al que únicamente el rol Admin puede tener acceso, por lo tanto, el código de la clase queda:

```
package tracking
import …
@Secured(['ROLE_ADMIN'])
class TiemposController { … }
```

En cambio, si tomamos *HomeController*, al que tanto el rol Admin como rol User tienen acceso, queda de la siguiente forma:

```
package tracking
import …
@Secured(['ROLE_ADMIN','ROLE_USER'])
class HomeController { … }
```

Estos ejemplos fueron a nivel de clase, donde la autorización de acceso es igual para todos los métodos, pero también se puede hacer autorización de acceso por cada método. En el proyecto Tracking hay un sólo controlador que comparte a dos módulos diferentes que es *PedidoController*, ya que tiene métodos que corresponden al módulo *Pedidos* y otros que corresponden al módulo de *Estadísticas*. Ya que *Pedidos* y *Estadísticas* tienen diferentes niveles de acceso (*Pedidos* es sólo para el rol Admin y Estadísticas para el rol Admin y rol User) se debe de configurar el acceso por método y no por clase. Queda entonces de la siguiente forma.

```
package tracking
import …
class PedidoController {
  @Secured(['ROLE_ADMIN'])
  def index(…) {…}
  @Secured(['ROLE_ADMIN'])
  def show(…) {…}
  @Secured(['ROLE_ADMIN'])
  def create() {…}
  @Secured(['ROLE_ADMIN'])
  def save(…) {…}
  @Secured(['ROLE_ADMIN'])
  def edit(…) {…}
  @Secured(['ROLE_ADMIN'])
  def update(…) {…}
  @Secured(['ROLE_ADMIN','ROLE_USER'])
  def resumen() {…}
  @Secured(['ROLE_ADMIN','ROLE_USER'])
  def getPedido() {…}
}
```

Entonces según la Tabla 2 y la Tabla 24 se obtiene la siguiente relación de acceso a controladores de acuerdo a la etiqueta @Secured (Tabla 25).

Acceso	Controladores
@Secured(['ROLE_ADMIN'])	VendedorController, ClienteController, ProveedorController, LineasController, TiemposController, PedidoController (métodos: index, show, create, save, edit, update)
@Secured(['ROLE_ADMIN','ROLE_USER'])	TrackingController, HomeController, PedidoController (métodos: resumen, getPedido)

Tabla 25. Acceso a controladores de cada rol.

Spring Security también utiliza control de acceso sobre las vistas, de tal forma que según el usuario que entra puede ver ciertos componentes. Para esto se vale de su biblioteca de etiquetas (SecurityTagLib), en la cual se pueden ver funcionalidades como:

- "*<sec:ifLoggedIn> </sec:ifLoggedIn>*" que mostrará lo que haya en su interior únicamente a los usuarios logueados.
- "*<sec:ifAnyGranted> </sec: ifAnyGranted>*" con su propiedad "roles=…" que mostrará su contenido a los usuarios logueados que pertenezcan a alguno de los roles nombrados

Si no se utiliza ninguna etiqueta la vista no hace ninguna acción y se muestra igual para todos los usuarios logueados. Como son únicamente dos niveles de permisos los que se manejan en este proyecto, entonces utilizo únicamente la etiqueta *ifAnyGranted* y así hago distinción entre lo que el rol Admin va a poder acceder (todo) y a lo que el rol User podrá acceder (algunos módulos).

Así en el layout *admin* (que springSecurityUI ejecuta después de que un usuario se loguea) se va a agregar la etiqueta:

```
<sec:ifAnyGranted roles="ROLE_ADMIN"> …
</sec:ifAnyGranted>
```

Y en medio de la etiqueta se agrega el contenido restringido. En este layout se encuentra el menú principal, y más adelante se verán los resultados.

Hay una tercera forma en que Spring Security puede autorizar acceso que se utiliza en el proyecto, que es agregando direcciones estáticas de la aplicación. Para esto se agregan dentro del archivo *Config* de la siguiente forma:

```
grails.plugin.springsecurity.controllerAnnotations.staticRules = [
   '/':                ['permitAll'],
   '/index':           ['permitAll'],
   '/index.gsp':       ['permitAll'],
   '/assets/**':       ['permitAll'],
   '/**/js/**':        ['permitAll'],
   '/**/css/**':       ['permitAll'],
   '/**/images/**':    ['permitAll'],
    '/plugins/jquery-ui-1.10.4/**':   ['permitAll'],
    '/**/materialadmin/**': ['permitAll'],
]
```

La anotación '*permitAll*' significa que es accesible para todos los roles y usuarios.

Vistas

El plugin de Spring Security automáticamente genera una ventana de login para acceder a la aplicación. (Ver Figura 27.)

3.5 Vendedores

Dominio

La funcionalidad del dominio viene dada por el siguiente código, que se representará en la base de datos.

```
package tracking
class Vendedor {
    String nombre
    String email
    static constraints = {
        nombre blank: false, nullable: false
        email blank: false, nullable: false, email: true, unique: true
    }
}
```

Se están creando dos atributos de la clase (*nombre*, *email*) los cuales se convierten en columnas de la tabla *vendedor* de la Base de Datos *tracking* (Figura 12).

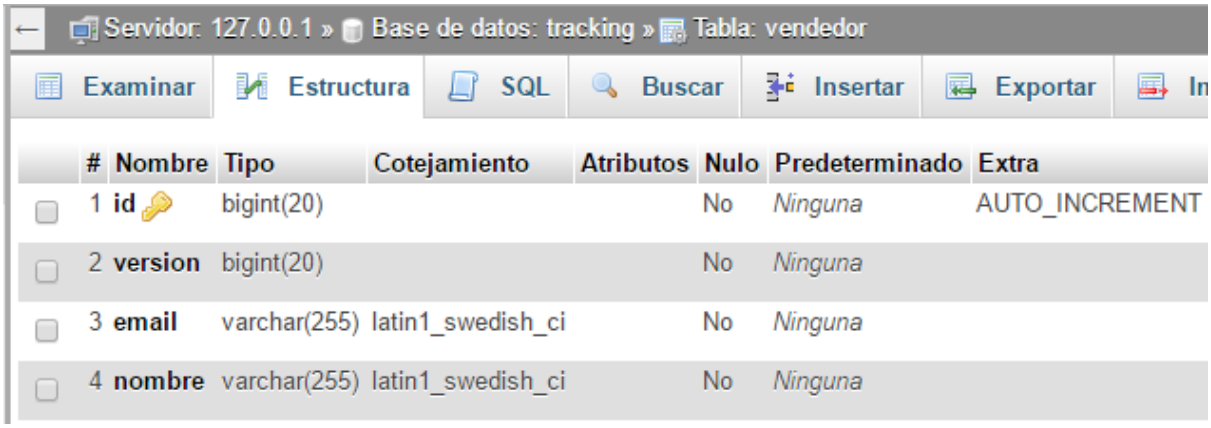


Figura 12. Datos de la tabla *vendedor*.

Automáticamente se crean siempre el *id* y la *version*. La configuración de cada atributo se agrega en el closure constraints, y los que se ocupan son:

- blank: si permite valores en blanco.
- nullable: si permite valores null.
- email: si tiene la estructura de un correo electrónico.
- unique: si es único y no se puede repetir ningún valor (como un id).

Controlador

A partir del dominio, Grails puede generar automáticamente el controlador *VendedorController* con métodos predeterminados (Tabla 26).

Método	Descripción	Parámetros	Retorno
index	Obtiene la lista de las entidades existentes.	max: Integer	Indefinido
show	Muestra las propiedades de una entidad seleccionada por medio de su identificador.	vendedorInstance: Vendedor	Indefinido
create	Genera una nueva entidad en el contexto de datos.	null	Indefinido

save	Persiste una nueva entidad en la base de datos.	vendedorInstance: Vendedor	Indefinido
edit	Obtiene una entidad existente.	vendedorInstance: Vendedor	Indefinido
update	Persiste una entidad existente en la base de datos.	vendedorInstance: Vendedor	Indefinido
delete	Elimina una entidad del contexto de datos.	vendedorInstance: Vendedor	Indefinido
notFound	Genera mensajes flash cada vez que se hace una acción sobre una entidad.	null	void

Tabla 26. Métodos del controlador de *Vendedores*.

Vistas

Las vistas se pueden generar a partir de los métodos del controlador, de tal manera que hay una vista por cada método (Tabla 27).

Vista	Descripción	Métodos del controlador
_form	Formularios para llenar los campos de la entidad.	N/A
create	Pantalla que muestra los formularios para crear una nueva entidad.	create
edit	Pantalla que muestra los formularios para editar una nueva entidad.	edit
index	Pantalla que muestra todas las entidades existentes.	index
show	Pantalla que muestra el detalle de una entidad determinada por su identificador.	show

Tabla 27. Vistas de *Vendedores*.

3.6 Clientes

Dominio

La funcionalidad del dominio viene dada por el siguiente código, que se representará en la base de datos.

```
package tracking
class Cliente {
    String nombre
    String email
    static constraints = {
        nombre blank: false, nullable: false
        email blank: false, nullable: false, unique: true, email: true
    }
}
```

Se están creando dos atributos de la clase (*nombre, email*) los cuales se convierten en columnas de la tabla *cliente* de la Base de Datos *tracking* (Figura 13).

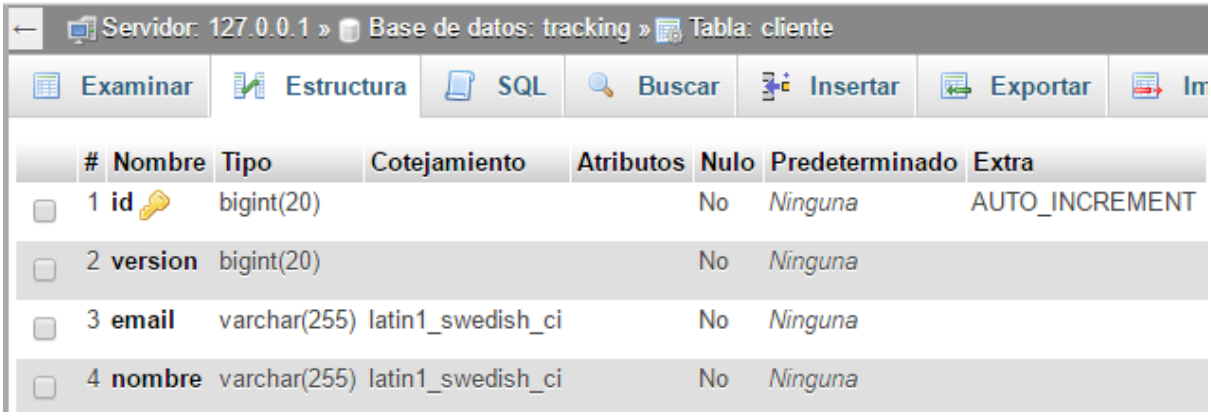


Figura 13. Datos de la tabla *cliente*.

Automáticamente se crean siempre el *id* y la *version*. La configuración de cada atributo se agrega en el closure *constraints*, y los que se ocupan son:

- blank: si permite valores en blanco.
- nullable: si permite valores null.
- email: si tiene la estructura de un correo electrónico.
- unique: si es único y no se puede repetir ningún valor (como un id).

Controlador

A partir del dominio, Grails puede generar automáticamente el controlador *ClienteController* con métodos predeterminados (Tabla 28).

Método	Descripción	Parámetros	Retorno
index	Obtiene la lista de las entidades existentes.	max: Integer	Indefinido
show	Muestra las propiedades de una entidad seleccionada por medio de su identificador.	clienteInstance: Cliente	Indefinido
create	Genera una nueva entidad en el contexto de datos.	null	Indefinido
save	Persiste una nueva entidad en la base de datos.	clienteInstance: Cliente	Indefinido
edit	Obtiene una entidad existente.	clienteInstance: Cliente	Indefinido
update	Persiste una entidad existente en la base de datos.	clienteInstance: Cliente	Indefinido
delete	Elimina una entidad del contexto de datos.	clienteInstance: Cliente	Indefinido
notFound	Genera mensajes flash cada vez que se hace una acción sobre una entidad.	null	void

Tabla 28. Métodos del controlador de *Clientes*.

Vistas

Las vistas se pueden generar a partir de los métodos del controlador, de tal manera que hay una vista por cada método (Tabla 29).

Vista	Descripción	Métodos del controlador
_form	Formularios para llenar los campos de la entidad.	N/A
create	Pantalla que muestra los formularios para crear una nueva entidad.	create
edit	Pantalla que muestra los formularios para editar una nueva entidad.	edit
index	Pantalla que muestra todas las entidades existentes.	index
show	Pantalla que muestra el detalle de una entidad determinada por su identificador.	show

Tabla 29. Vistas de *Clientes*.

3.7 Proveedores

Dominio

La funcionalidad del dominio viene dada por el siguiente código, que se representará en la base de datos.

```
package tracking
class Proveedor {
    String nombre
    static constraints = {
        nombre blank: false, nullable: false
    }
}
```

Se está creando un atributo de la clase (*nombre*) el cual se convierte en columna de la tabla *proovedor* de la Base de Datos *tracking* (Figura 14).

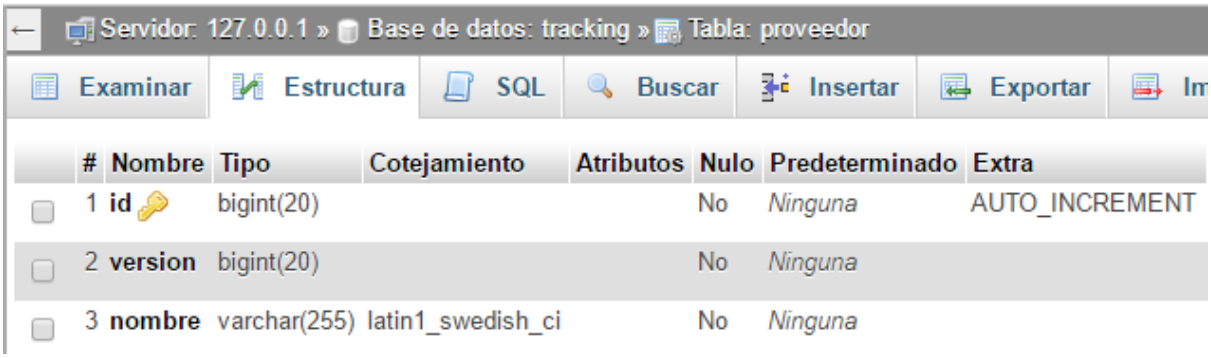


Figura 14. Datos de la tabla *proveedor*.

Automáticamente se van a crear siempre el *id* y la *version*. La configuración de cada atributo se agrega en el closure *constraints*, y los que se ocupan son:

- blank: si permite valores en blanco.
- nullable: si permite valores null.

Controlador

A partir del dominio, Grails puede generar automáticamente el controlador *ProveedorController* con métodos predeterminados (Tabla 30).

Método	Descripción	Parámetros	Retorno
index	Obtiene la lista de las entidades existentes.	max: Integer	Indefinido
show	Muestra las propiedades de una entidad seleccionada por medio de su identificador.	proveedorInstance: Proveedor	Indefinido
create	Genera una nueva entidad en el contexto de datos.	null	Indefinido
save	Persiste una nueva entidad en la base de datos.	proveedorInstance: Proveedor	Indefinido
edit	Obtiene una entidad existente.	proveedorInstance: Proveedor	Indefinido
update	Persiste una entidad existente en la base de datos.	proveedorInstance: Proveedor	Indefinido
delete	Elimina una entidad del contexto de datos.	proveedorInstance: Proveedor	Indefinido
notFound	Genera mensajes flash cada vez que se hace una acción sobre una entidad.	null	void

Tabla 30. Métodos del controlador de *Proveedores*.

Vistas

Las vistas se pueden generar a partir de los métodos del controlador, de tal manera que hay una vista por cada método (Tabla 31).

Vista	Descripción	Métodos del controlador
_form	Formularios para llenar los campos de la entidad.	N/A
create	Pantalla que muestra los formularios para crear una nueva entidad.	create
edit	Pantalla que muestra los formularios para editar una nueva entidad.	edit
index	Pantalla que muestra todas las entidades existentes.	index
show	Pantalla que muestra el detalle de una entidad determinada por su identificador.	show

Tabla 31. Vistas de *Proveedores*.

3.8 Líneas de Negocio

Dominio

La funcionalidad del dominio viene dada por el siguiente código, que se representará en la base de datos.

```
package tracking
class Lineas {
    String nombre
    static constraints = {
        nombre blank: false, nullable: false
    }
}
```

Se está creando un atributo de la clase (*nombre*) el cual se convierte en columna de la tabla *lineas* de la Base de Datos *tracking* (Figura 15).

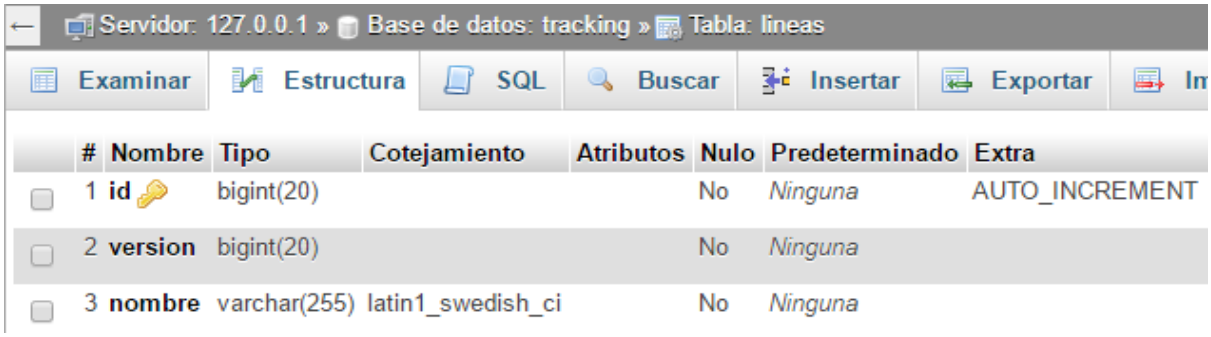


Figura 15. Datos de la tabla *lineas*.

Automáticamente se van a crear siempre el *id* y la *version*. La configuración de cada atributo se agrega en el closure 'constraints', y los que se ocupan son:

- blank: si permite valores en blanco.
- nullable: si permite valores null.

Controlador

A partir del dominio, Grails puede generar automáticamente el controlador *LineasController* con métodos predeterminados (Tabla 32).

Método	Descripción	Parámetros	Retorno
index	Obtiene la lista de las entidades existentes.	max: Integer	Indefinido
show	Muestra las propiedades de una entidad seleccionada por medio de su identificador.	lineasInstance: Lineas	Indefinido
create	Genera una nueva entidad en el contexto de datos.	null	Indefinido
save	Persiste una nueva entidad en la base de datos.	lineasInstance: Lineas	Indefinido
edit	Obtiene una entidad existente.	lineasInstance: Lineas	Indefinido
update	Persiste una entidad existente en la base de datos.	lineasInstance: Lineas	Indefinido

delete	Elimina una entidad del contexto de datos.	lineasInstance: Lineas	Indefinido
notFound	Genera mensajes flash cada vez que se hace una acción sobre una entidad.	null	void

Tabla 32. Métodos del controlador de *Líneas de Negocio*.

Vistas

Las vistas se pueden generar a partir de los métodos del controlador, de tal manera que hay una vista por cada método (Tabla 33).

Vista	Descripción	Métodos del controlador
_form	Formularios para llenar los campos de la entidad.	N/A
create	Pantalla que muestra los formularios para crear una nueva entidad.	create
edit	Pantalla que muestra los formularios para editar una nueva entidad.	edit
index	Pantalla que muestra todas las entidades existentes.	index
show	Pantalla que muestra el detalle de una entidad determinada por su identificador.	show

Tabla 33. Vistas de *Líneas de Negocio*.

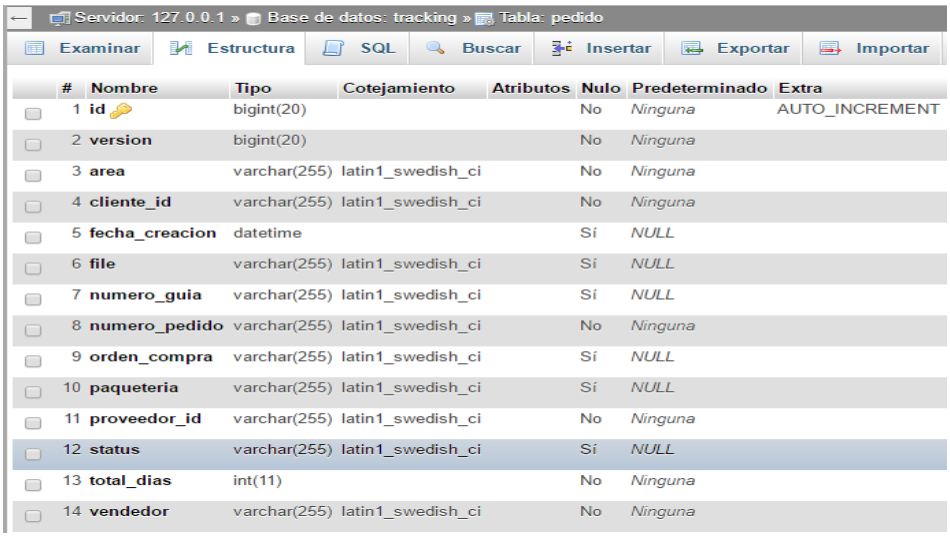
3.9 Pedidos

Dominio

La funcionalidad del dominio viene dada por el siguiente código, que se representará en la base de datos.

```
package tracking
class Pedido {
    String numero_pedido
    String orden_compra
    String file
    String vendedor
    String area
    String cliente_id
    String paqueteria
    String numero_guia
    String proveedor_id
    String status
    int total_dias
    Date fecha_creacion
    static constraints = {
        numero_pedido blank: false, nullable: false, unique: true
        orden_compra blank: true, nullable: true
        file blank: true, nullable: true
        vendedor blank: false, nullable: false
        area blank: false, nullable: false
        cliente_id blank: false, nullable: false
        paqueteria blank: true, nullable: true
        numero_guia blank: true, nullable: true
        status blank: true, nullable: true
        total_dias blank: true, nullable: true
        fecha_creacion blank: true, nullable: true
        proveedor_id blank: false, nullable: false
    }
}
```

Se están creando doce atributos de la clase (*numero_pedido*, *orden_compra*, *file*, *vendedor*, *area*, *cliente_*id, *paqueteria*, *numero_guia*, *proveedor_id*, *status*, *total_dias*, *fecha_creacion*) los cuales se convierten en columnas de la tabla *pedido* de la Base de Datos *tracking* (Figura 16).



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	bigint(20)			No	*Ninguna*	AUTO_INCREMENT
2	version	bigint(20)			No	*Ninguna*	
3	area	varchar(255)	latin1_swedish_ci		No	*Ninguna*	
4	cliente_id	varchar(255)	latin1_swedish_ci		No	*Ninguna*	
5	fecha_creacion	datetime			Sí	*NULL*	
6	file	varchar(255)	latin1_swedish_ci		Sí	*NULL*	
7	numero_guia	varchar(255)	latin1_swedish_ci		Sí	*NULL*	
8	numero_pedido	varchar(255)	latin1_swedish_ci		No	*Ninguna*	
9	orden_compra	varchar(255)	latin1_swedish_ci		Sí	*NULL*	
10	paqueteria	varchar(255)	latin1_swedish_ci		Sí	*NULL*	
11	proveedor_id	varchar(255)	latin1_swedish_ci		No	*Ninguna*	
12	status	varchar(255)	latin1_swedish_ci		Sí	*NULL*	
13	total_dias	int(11)			No	*Ninguna*	
14	vendedor	varchar(255)	latin1_swedish_ci		No	*Ninguna*	

Figura 16. Datos de la tabla *pedido*.

Automáticamente se van a crear siempre el *id* y la *version*. La configuración de cada atributo se agrega en el closure *constraints*, y las que se ocupan significan lo siguiente:

- blank: si permite valores en blanco.
- nullable: si permite valores null.
- unique: si es único y no se puede repetir ningún valor (como un id).

Controlador

El controlador *PedidoController* contiene métodos del módulo Pedidos y del módulo Estadísticas. Los métodos del módulo Pedidos son los siguientes (Tabla 34).

Método	Descripción	Parámetros	Retorno
index	Obtiene la lista de las entidades existentes.	max: Integer	Indefinido
show	Muestra las propiedades de una entidad seleccionada por medio de su identificador.	pedidoInstance: Pedido	Indefinido
create	Genera una nueva entidad en el contexto de datos.	null	Indefinido
save	Persiste una nueva entidad en la base de datos.	pedidoInstance: Pedido	Indefinido
edit	Obtiene una entidad existente.	pedidoInstance: Pedido	Indefinido
update	Persiste una entidad existente en la base de datos.	pedidoInstance: Pedido	Indefinido
delete	Elimina una entidad del contexto de datos.	pedidoInstance: Pedido	Indefinido
notFound	Genera mensajes flash cada vez que se hace una acción sobre una entidad.	null	void

Tabla 34. Métodos del controlador de *Pedidos*.

Servicio

El servicio *PedidoService* tiene un método que es utilizado por el controlador *PedidoController* en sus métodos *save* y *update* (Tabla 35).

Método	Descripción	Parámetros	Retorno
updateFileToFolder	Actualiza un archivo en un directorio. Recibe la referencia al archivo seleccionado para cargar al sistema el nombre con el cual renombrar el archivo y el directorio dónde guardarlo. La carpeta raíz para guardar los archivos es '*web-app*'. En caso de que no exista la carpeta la crea.	file: MultipartFile, name: String, directory: String	Indefinido

Tabla 35. Métodos del servicio *PedidoService* de *Pedidos*.

El servicio *TrackingService* tiene un método que es utilizado por el controlador *PedidoController* en su método *index* (Tabla 36).

Método	Descripción	Parámetros	Retorno
etapaActualAll	Obtiene una lista de todos los pedidos en su última etapa; cada pedido con su etapa última, número de orden, vendedor, cliente, estatus y archivo adjunto correspondiente a la etapa.	null	List<Expando>

Tabla 36. Métodos del servicio *TrackingService* de *Pedidos*.

Vistas

Las vistas se pueden generar a partir de los métodos del controlador, de tal manera que hay una vista por cada método (Tabla 37).

Vista	Descripción	Métodos del controlador
_form	Formularios para llenar los campos de la entidad.	N/A
create	Pantalla que muestra los formularios para crear una nueva entidad.	create
edit	Pantalla que muestra los formularios para editar una nueva entidad.	edit
index	Pantalla que muestra todas las entidades existentes.	index
show	Pantalla que muestra el detalle de una entidad determinada por su identificador.	show

Tabla 37. Vistas de *Pedidos*.

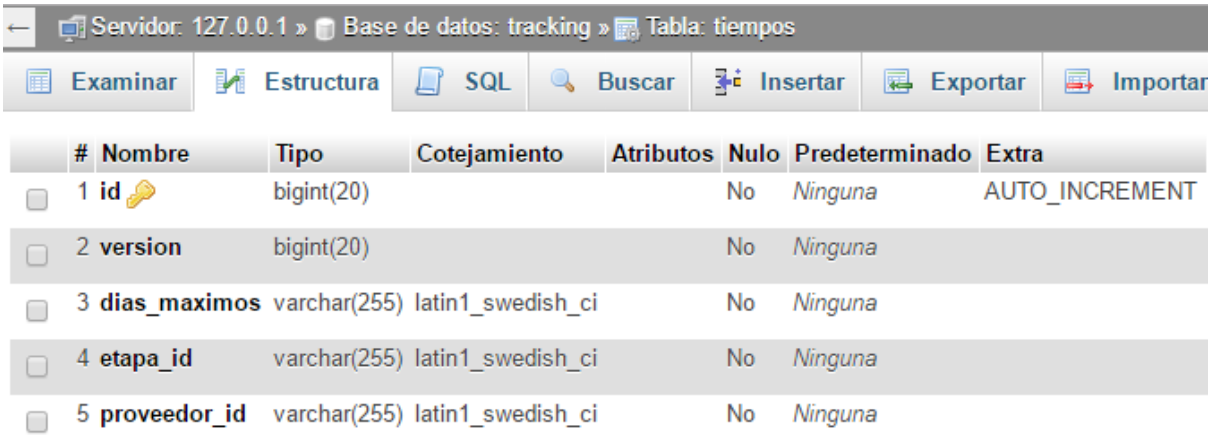
3.10 Tiempos

Dominio

La funcionalidad del dominio viene dada por el siguiente código, que se representará en la base de datos.

```
package tracking
class Tiempos {
    String proveedor_id
    String etapa_id
    String dias_maximos
    static constraints = {
        proveedor_id blank: false, nullable: false
        etapa_id blank: false, nullable: false
        dias_maximos blank: false, nullable: false
    }
}
```

Se están creando tres atributos de la clase (*proveedor_id*, *etapa_id*, *dias_maximos*) los cuales se convierten en columnas de la tabla *tiempos* de la Base de Datos *tracking* (Figura 17).



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	bigint(20)			No	*Ninguna*	AUTO_INCREMENT
2	version	bigint(20)			No	*Ninguna*	
3	dias_maximos	varchar(255)	latin1_swedish_ci		No	*Ninguna*	
4	etapa_id	varchar(255)	latin1_swedish_ci		No	*Ninguna*	
5	proveedor_id	varchar(255)	latin1_swedish_ci		No	*Ninguna*	

Figura 17. Datos de la tabla *tiempos*.

Automáticamente se van a crear siempre el *id* y la *version*. La configuración de cada atributo se agrega en el closure *constraints*, y los que se ocupan son:

- blank: si permite valores en blanco.
- nullable: si permite valores null.

Controlador

El controlador *TiemposController* contiene los siguientes métodos (Tabla 38).

Método	Descripción	Parámetros	Retorno
index	Obtiene la lista de las entidades existentes asignadas por etapa a cada proveedor.	max: Integer	Indefinido
show	Muestra las entidades que le corresponden a cierto proveedor, por medio del id del proveedor.	pedidoInstance: Pedido	Indefinido

create	Genera siete entidades nuevas, cada una le corresponde a una etapa diferente y a un mismo proveedor.	null	Indefinido
save	Persiste siete entidades en la base de datos.	pedidoInstance: Pedido	Indefinido
edit	Obtiene una entidad existente.	pedidoInstance: Pedido	Indefinido
update	Persiste una entidad existente en la base de datos.	pedidoInstance: Pedido	Indefinido
notFound	Genera mensajes flash cada vez que se hace una acción sobre una entidad.	null	void

Tabla 38. Métodos del controlador de *Tiempos*.

Servicio

El servicio *TiemposService* tiene los siguientes métodos (Tabla 39) utilizados por *TiemposController* en sus métodos *index*, *show*, *create* y *edit*.

Método	Descripción	Parámetros	Retorno
getProveedores	Obtiene todos los proveedores que aún no tengan tiempo asociados.	null	List<Expando>
getTiemposBy Proveedor	Obtiene la etapa y el tiempo de cierto proveedor.	proveedorInstance: Proveedor	List<Expando>
getTiempoAll Proveedor	Obtiene la etapa y el tiempo de todos los proveedores que tienen tiempos asignados.	null	List<Expando>
etapaProveedor	Obtiene la etapa y el proveedor de cierto registro de tiempo	tiemposInstance: Tiempos	List<Expando>

Tabla 39. Métodos del servicio de *Tiempos*.

Vistas

Las vistas son las siguientes (Tabla 40).

Vista	Descripción	Métodos del controlador
_form	Formularios para llenar los campos de la entidad.	N/A
create	Pantalla que muestra los formularios para crear una nueva entidad.	create
edit	Pantalla que muestra los formularios para editar una nueva entidad.	edit
index	Pantalla que muestra todas las entidades existentes.	index
show	Pantalla que muestra el detalle de una entidad determinada por su identificador.	show

Tabla 40. Vistas de *Tiempos*.

3.11 Administración

Dominio

Este módulo es más complejo que los anteriores. Está conformado por dos dominios. La funcionalidad del primer dominio viene dada por el siguiente código, que se representará en la base de datos.

```
package tracking
class Etapa {
    String nombre
    static constraints = {
        nombre blank:false, nullable: false
    }
}
```

Se está creando un atributo de la clase (*nombre*) el cual se convierte en columna de la tabla *etapa* de la Base de Datos *tracking* (Figura 18).

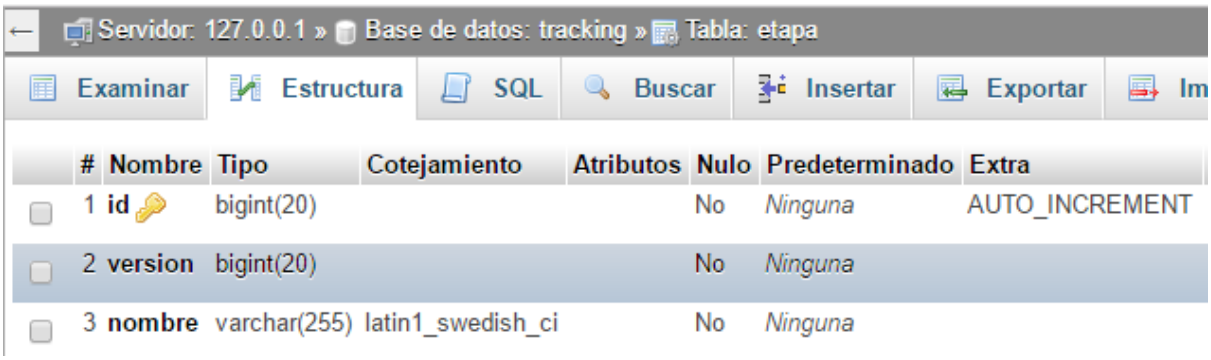


Figura 18. Datos de la tabla *etapa*.

Automáticamente se van a crear siempre el *id* y la *version*. La configuración de cada atributo se agrega en el closure *constraints*, y las que se ocupan significan lo siguiente:

- blank: si permite valores en blanco.
- nullable: si permite valores null.

Esta tabla juega el papel de un catálogo, el cual no se va a modificar durante el uso de la aplicación, únicamente se va a consultar. Por lo tanto, se va llenar una única vez y quedará almacenado en la base de datos. Para esto se agrega en el archivo *BootStrap* dentro del closure *init* el siguiente código:

```
if(Etapa.count()==0){
    new Etapa(nombre:'Pedido generado').save(flush: true)
    new Etapa(nombre:'Orden de compra generada').save(flush: true)
    new Etapa(nombre:'Embarque').save(flush: true)
    new Etapa(nombre:'Arribo').save(flush: true)
    new Etapa(nombre:'Revision').save(flush: true)
    new Etapa(nombre:'Envio').save(flush: true)
    new Etapa(nombre:'Recepcion').save(flush: true)
}
```

Entonces el id y nombre que le corresponden a cada etapa del pedido son los siguiente (se omitieron los acentos para evitar cualquier problema de formato):

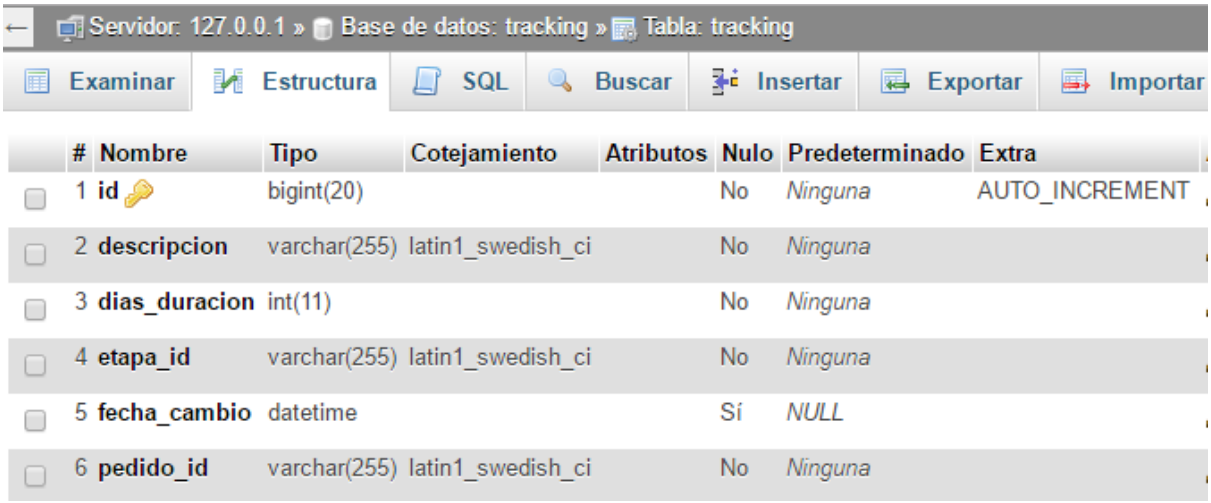
1. Pedido generado
2. Orden
3. Embarque
4. Arribo

5. Revision
6. Envio
7. Recepcion

La funcionalidad del segundo dominio viene dada por el siguiente código, que se representará en la base de datos.

```
package tracking
class Tracking {
    String pedidoId
    String etapaId
    Date fechaCambio
    String descripcion
    int diasDuracion
    static constraints = {
        pedidoId blank: false, nullable:false
        etapaId blank:false, nullable: false
        fechaCambio blank: true, nullable: true
        descripcion blank: false, nullable: false
        diasDuracion blank: true, nullable: true
    }
    static mapping = {version false}
}
```

Se están creando cinco atributos de la clase (*pedidoId*, *etapaId*, *fechaCambio*, *descripcion*, *diasDuracion*) los cuales se convierten en columnas de la tabla *tracking* de la Base de Datos *tracking* (Figura 19).



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	bigint(20)			No	Ninguna	AUTO_INCREMENT
2	descripcion	varchar(255)	latin1_swedish_ci		No	Ninguna	
3	dias_duracion	int(11)			No	Ninguna	
4	etapa_id	varchar(255)	latin1_swedish_ci		No	Ninguna	
5	fecha_cambio	datetime			Sí	NULL	
6	pedido_id	varchar(255)	latin1_swedish_ci		No	Ninguna	

Figura 19. Datos de la tabla *tracking*.

Automáticamente se van a crear siempre el *id* y la *version*. La configuración de cada atributo se agrega en el closure *constraints*, y las que se ocupan significan lo siguiente:

- blank: si permite valores en blanco.
- nullable: si permite valores null.

Sin embargo, se ve a un atributo *mapping* el cual tiene un atributo *version* que se le asigna el valor *false*. Esto hará que la tabla no genere versión, tal como se ve en la tabla tracking. (Figura 20.) Se hace esto porque en la implementación de los métodos del controlador el guardado en la base de dato se hace de forma manual (ver más adelante los métodos del Servicio), y eso no lo permitiría el gestor de MySQL si existiesen varias versiones de la tabla, dada la posibilidad de conflictos con las versiones. Sin embargo, en la lógica de la implementación se evitan posibles conflictos con las versiones de las instancias.

Controlador

El controlador *TrackingController* contiene los siguientes métodos (Tabla 41).

Método	Descripción	Parámetros	Retorno
ordenDeCompra	genera una nueva orden de compra para un pedido.	null	Indefinido
saveOrden	persiste en la base de datos una orden de compra.	trackingInstance: Tracking	Indefinido
embarque	genera un embarque para un pedido.	null	Indefinido
saveEmbarque	verifica que se pueda procesar el embarque y en su caso lo persiste en la base de datos.	trackingInstance: Tracking	Indefinido
verificaOrdenListaParaEmbarque	valida que todos los pedidos asociados a cierto número de orden ya estén listos para el embarque.	numeroOrden: String	aux: Bool
verificaFechas	valida que todas las fechas asociadas a cierto número de orden sean menores que cierta fecha.	numeroOrden: String, fecha: Date	aux: Bool
arribo	genera un arribo para un pedido.	null	Indefinido
saveArribo	verifica que se pueda procesar el arribo y en su caso lo persiste en la base de datos.	trackingInstance: Tracking	Indefinido
revision	genera una revisión para su pedido.	null	Indefinido
saveRevision	verifica que se pueda procesar la revisión y en su caso lo persiste en a base de datos.	trackingInstance: Tracking	Indefinido
envio	genera un envio para su pedido.	null	Indefinido
saveEnvio	verifica que se pueda procesar el envio y en su caso lo persiste en la base de datos.	trackingInstance: Tracking	Indefinido
recepcion	genera una recepción para su pedido.	null	Indefinido
saveRecepcion	verifica que se pueda procesar el envio y en su caso lo persiste, así como lo días totales del pedido, en la base de datos.	trackingInstance: Tracking	Indefinido

Tabla 41. Métodos del controlador *TrackingController* de *Administración*.

Servicio

El servicio *TrackingService* tiene los siguientes métodos (Tabla 42) utilizados por *TrackingController* en su método *save*.

Método	Descripción	Parámetros	Retorno
Create	Crea un registro en la tabla *tracking*. Dada la lógica de negocio, siempre será uno nuevo.	pedidoId: Long, etapaId: String, fechaCambio: Date, descripcion: String	Indefinido

Tabla 42. Métodos del servicio *TrackingService* de *Administración*.

Vistas

Las vistas son las siguientes (Tabla 43).

Vista	Descripción	Métodos del controlador
ordenDeCompra	Pantalla que muestra los detalles del avance del pedido a la etapa 2. Orden.	ordenDeCompra
embarque	Pantalla que muestra los detalles del avance del pedido a la etapa 3. Embarque.	embarque
arribo	Pantalla que muestra los detalles del avance del pedido a la etapa 4. Arribo.	arribo
revision	Pantalla que muestra los detalles del avance del pedido a la etapa 5. Revision.	revision
envio	Pantalla que muestra los detalles del avance del pedido a la etapa 6. Envio.	envio
recepcion	Pantalla que muestra los detalles del avance del pedido a la etapa 7. Recepcion.	recepcion

Tabla 43. Vistas de *Administración*.

3.12 Estadísticas

Dominio

Este módulo no cuenta con dominio.

Controlador

El controlador *PedidoController* contiene métodos del módulo Pedidos y del módulo Estadísticas. Los métodos del módulo Estadísticas son los siguientes (Tabla 44).

Método	Descripción	Parámetros	Retorno
getPedido	Muestra los detalles del ciclo de vida de cierto pedido.	null	Indefinido
resume	Genera una nueva ventana para elegir el pedido del cual se quiera ver el resumen de sus movimientos.	null	Indefinido

Tabla 44. Métodos del controlador *PedidoController* de *Estadísticas*.

El controlador *HomeController* contiene los siguientes métodos (Tabla 45).

Método	Descripción	Parámetros	Retorno
index	Muestra el *Home* de la aplicación, donde vienen desglosadas todas las estadísticas (gráfica de pedidos, movimientos recientes, total de pedidos, ventas por departamento, pedidos por cliente).	null	Indefinido
pedidoEtapa	Muestra todos los pedidos que están asociados con una etapa seleccionada.	etapaInstance: Etapa	Indefinido
pedidoCliente	Muestra todos los pedidos que están asociados con un cliente seleccionado.	clienteInstance: Cliente	Indefinido

Tabla 45. Métodos del controlador *HomeController* de *Estadísticas*.

Servicio

El servicio *ClienteService* tiene los siguientes métodos (Tabla 46) utilizados por *HomeController* en sus métodos *index* y *pedidoCliente*.

Método	Descripción	Parámetros	Retorno
topClientes	Obtiene los N clientes que han hecho más pedidos en el año y la cantidad de pedidos.	limit: int	List<Expando>
pedidosCliente	Obtiene todos los pedidos que tiene cierto cliente, del año en curso.	clienteInstance: Cliente	List<Expando>

Tabla 46. Métodos del servicio *ClienteService* de *Estadísticas*.

El servicio *RastreoService* tiene los siguientes métodos (Tabla 47) utilizados por *HomeController* en su método *getPedido*.

Método	Descripción	Parámetros	Retorno
getResumenDe Pedido	Obtiene una lista con la fecha, días transcurridos, días máximos y días excedidos por cada etapa, de cierto pedido.	pedido: String	List<Expando>
getDuracionMaxima	Obtiene una lista de los días máximos de cada etapa, de cierto proveedor.	proveedor: String	List<Expando>
getDiasTotales	Devuelve la suma de los días reales que tiene el pedido desde que se generó hasta el momento.	pedido: String	List<Expando>
getDiasMaximos	Devuelve la suma de los días máximos de cada etapa, de cierto proveedor.	proveedor: String	List<Expando>
getPedidosPorMes	Devuelve una lista con el total de pedidos de cada mes.	null	List<Expando>

Tabla 47. Métodos del servicio *RastreoService* de *Estadísticas*.

El servicio *TrackingService* tiene los siguientes métodos (Tabla 48) utilizados por *HomeController* en sus métodos *index* y *pedidosEtapa*.

Método	Descripción	Parámetros	Retorno
tablaIndex	Obtiene los últimos 10 movimientos hechos en el sistema.	null	List<Expando>
ventasPor Departamento	Obtiene el total de pedidos que tiene cada departamento y el total de pedidos del sistema; todo esto en lo que va del año.	null	List<Expando>
totalEtapas	Obtiene el total de registros que tienen cada etapa en la tabla tracking.	etapa: int, nombre: String	List<Expando>
pedidoEtapas	Obtiene todos los pedidos que estén en cierta etapa; en el año en curso.	etapaInstance: Etapa	List<Expando>

Tabla 48. Métodos del servicio *TrackingService* de *Estadísticas*.

Vistas

Las vistas son las siguientes (Tabla 49).

Vista	Descripción	Métodos del controlador
pedido/_pedido	Pantalla parcial que se integra dentro de la pantalla pedido/resumen que muestra el detalle del ciclo de vida de un cierto pedido.	PedidoController/ getPedido
pedido/resumen	Pantalla que permite elegir el pedido del cual se quiera ver el resumen de sus movimientos.	PedidoController/ resume
home/index	Pantalla de Home de la aplicación. Muestra gráficas y estadísticas del sistema.	HomeController/ index
home/pedidosClientes	Pantalla que muestra todos los pedidos que están asociados con un cliente seleccionada.	HomeController/ pedidoCliente
home/pedidosEtapa	Pantalla que muestra todos los pedidos que están asociados con una etapa seleccionada.	HomeController/ pedidoEtapa

Tabla 49. Vistas de *Estadísticas*.

Capítulo 4. Producción

En este capítulo se verá la forma en que la aplicación Tracking se lleva a producción, es decir, cuando se concluye el desarrollo y se entrega como producto al cliente; a este proceso también se le conoce como puesta en producción. Además, se mostrará a detalle el flujo de cada uno de los módulos de la aplicación, mostrando su uso real con ejemplos prácticos.

4.1 Puesta en producción

Lo que se debe de hacer es crear el archivo WAR (Web Application Archive) de la aplicación Tracking y luego agregarlo a Tomcat, con sus respectivas configuraciones, para que se pueda utilizar la aplicación como producto final.

Primero se genera el archivo WAR. Se puede hacer desde IntelliJ IDEA o con Grails. Opté por utilizar los comandos que Grails proporciona para este propósito (http://docs.grails.org/2.1.0/ref/Command%20Line/war.html) utilizando la consola del IDE.

Se escribe **grails war** en la consola (Figura 20).

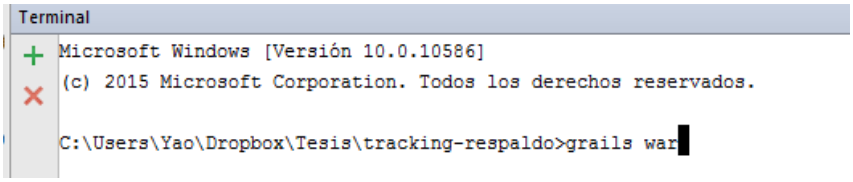


Figura 20. Consola de comandos de Intellij IDEA.

Cuando finaliza se genera un archivo *tracking.war* en la carpeta *web-app* del proyecto. Este archivo se mueve a la carpeta *xampp\tomcat\webapps*.

A continuación, se debe de reiniciar Tomcat desde la consola de XAMPP (Figura 21).

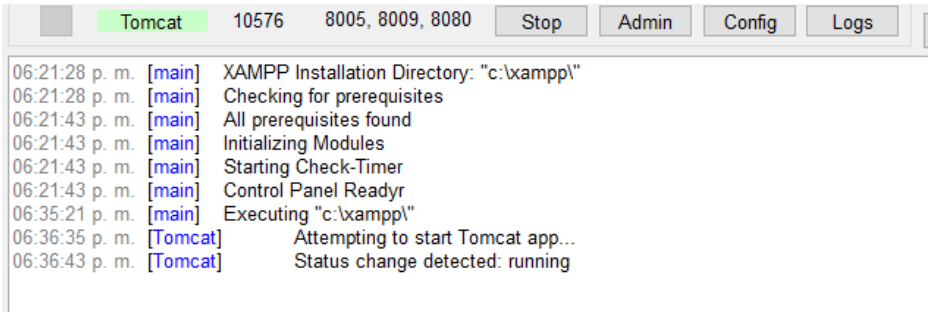


Figura 21. Consola de comandos de XAMPP.

Ahora al acceder a localhost (http://localhost:8080/tracking) se puede ver la aplicación desplegada.

4.2 Flujo de trabajo

A continuación, se va a mostrar el flujo completo de trabajo de toda la aplicación ya totalmente terminada y funcional.

4.2.1 Login

- Al abrir la aplicación se verá la ventana para ingresar por medio de un login (/tracking/login/auth). No se podrá pasar de esta ventana a menos que se ingresen credenciales correctas (Figura 22).

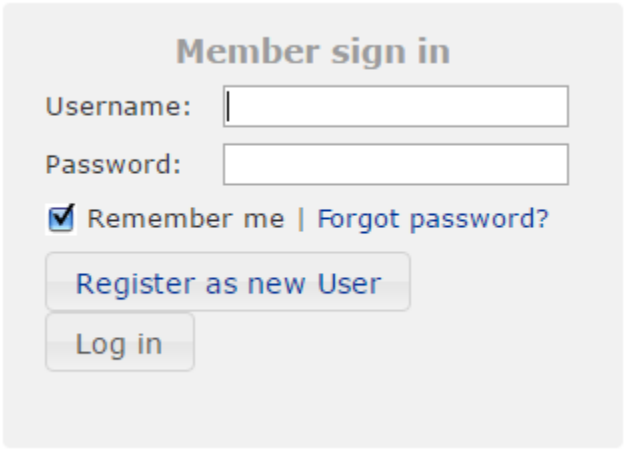


Figura 22. Pantalla inicial de Login.

- Se debe de ingresar un nombre de usuario y contraseña. En caso de que el nombre de usuario no exista o la contraseña sea incorrecta aparecerá una leyenda que así lo indique. La opción de “recordar” (Remember me) se puede desactivar (Figura 23).

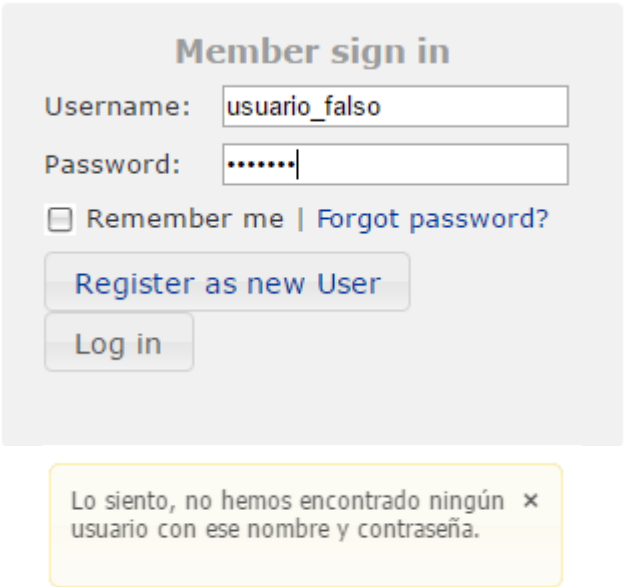


Figura 23. Pantalla de error de Login.

- Las opciones de *recuperar contraseña* (Forgor password?) y *registrar nuevo usuario* (Register as new User) no están habilitadas.
- Una vez que se ingresen credenciales correctas se abrirá el *Home* de la aplicación con el nombre del usuario en la esquina superior izquierda (Figura 24).

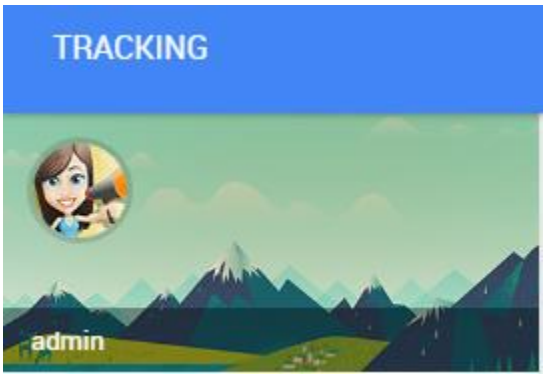


Figura 24. Sección de opciones generales.

- Para hacer logout se le selecciona al nombre de usuario y se despliega dicha opción (Figura 25).

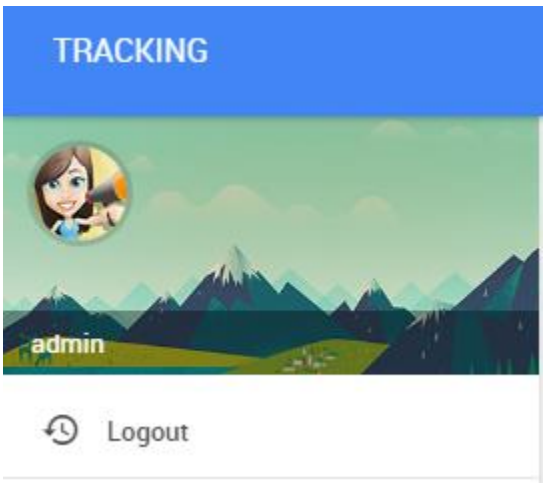


Figura 25. Opción de Logout.

- Las opciones que el rol Administrador se muestran en la Figura 26.

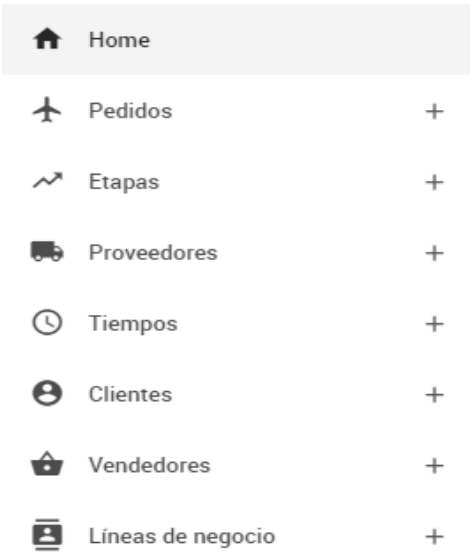


Figura 26. Menú principal rol Administrador.

- Las opciones que el rol Usuario ve muestran en la Figura 27.

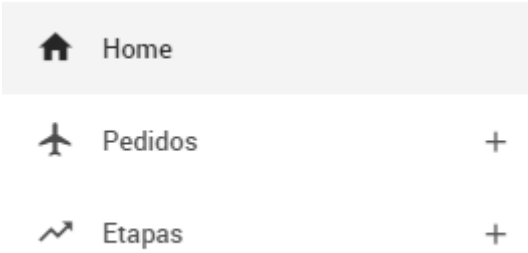


Figura 27. Menú principal rol Usuario.

4.2.2 Vendedores

- Únicamente se puede acceder a este módulo con rol Administrador y en el menú del *Home*. (Figura 28)

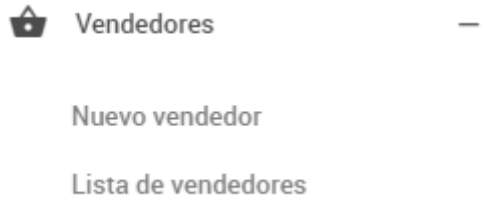


Figura 28. Menú de vendedores.

- Para crear un nuevo vendedor se selecciona la opción *Nuevo vendedor* y se abre una nueva ventana (/tracking/vendedor/create) (Figura 29).

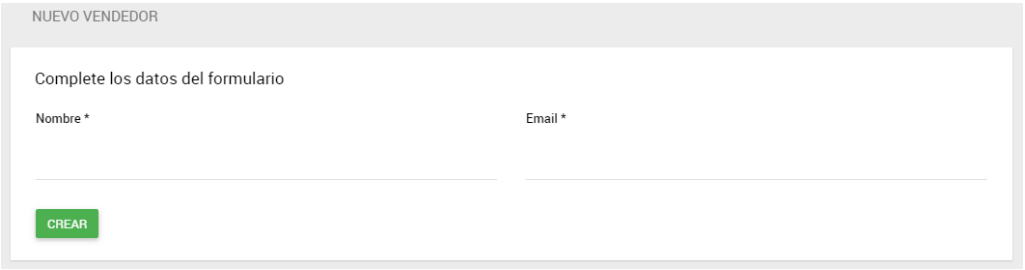


Figura 29. Nuevo de vendedor.

- Si se intenta crear un vendedor con un email que ya exista en la base de datos el sistema arroja un error (Figura 30).

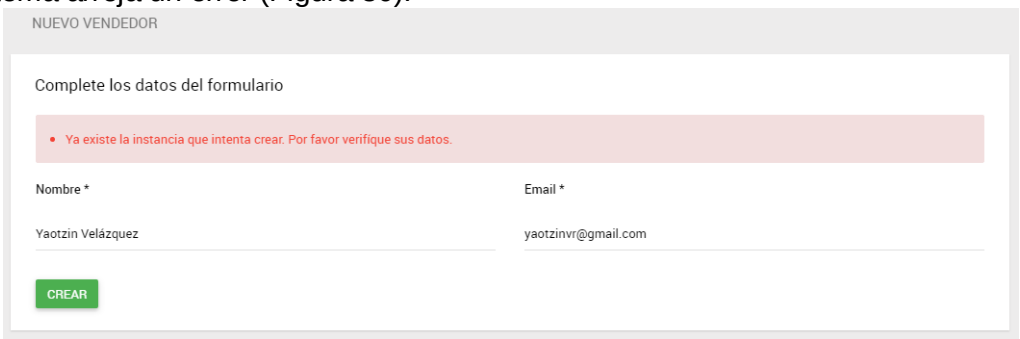


Figura 30. Error de nombre para nuevo de vendedor.

- Si se intenta registrar un email que no sea válido el sistema arroja un error (Figura 31).

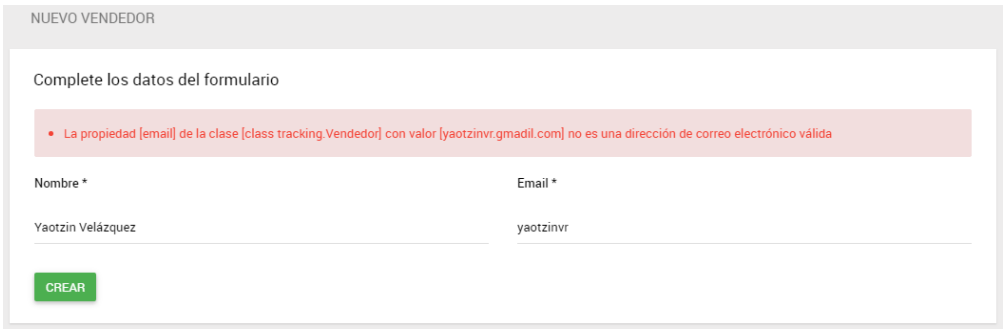


Figura 31. Error de email para nuevo de vendedor.

- Al crearse correctamente un vendedor se ve la siguiente pantalla (Figura 32).

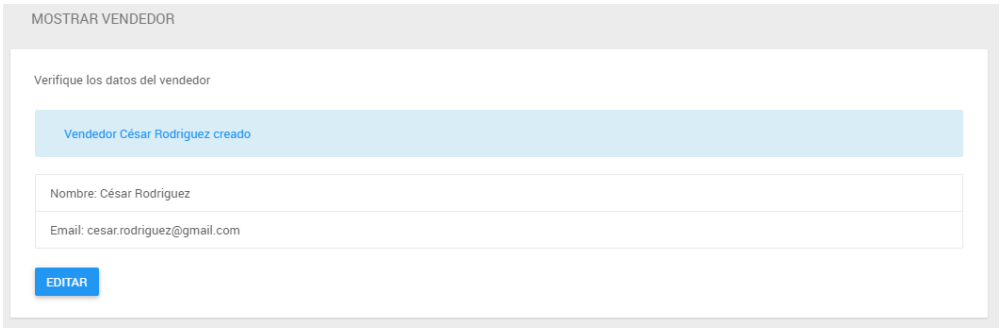


Figura 32. Vendedor creado correctamente.

- Se puede listar a todos los vendedores existentes (/tracking/vendedor/index) (Figura 33).

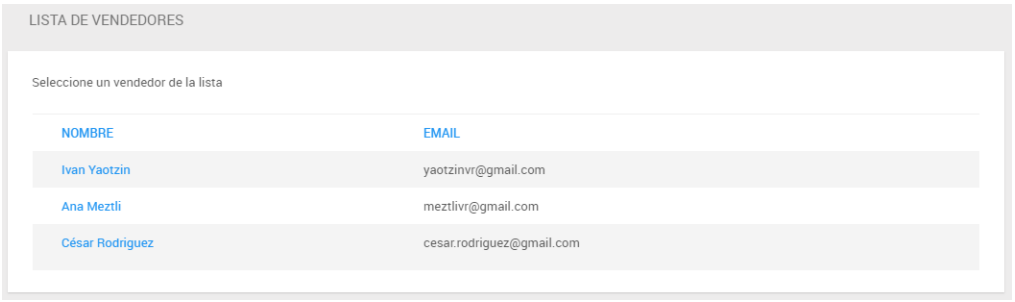


Figura 33. Lista de vendedores.

- Para ver los detalles de cada uno se selecciona el nombre y se abre una nueva ventana (/tracking/vendedor/show/{id}) (Figura 34).

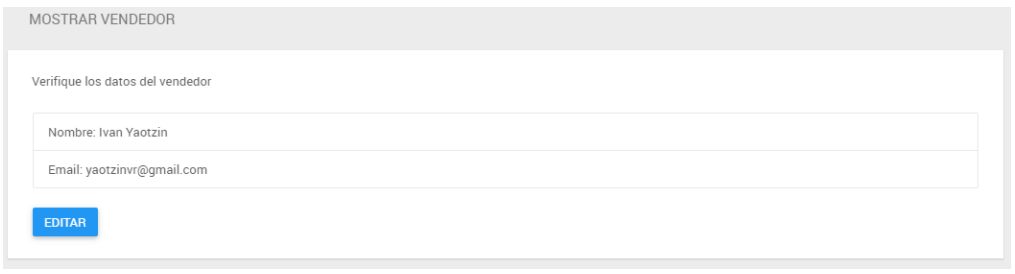


Figura 34. Detalle de vendedor.

- Sólo desde esta ventana se tiene acceso a editar el vendedor (/tracking/vendedor/edit/{id}) (Figura 35).

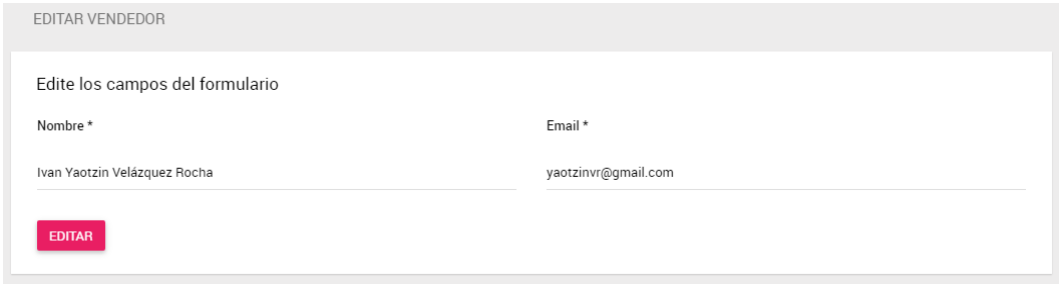


Figura 35. Editar vendedor.

- Al editar se tienen las mismas reglas que al crear uno nuevo. Al actualizarlo correctamente se ve la siguiente ventana (Figura 36).

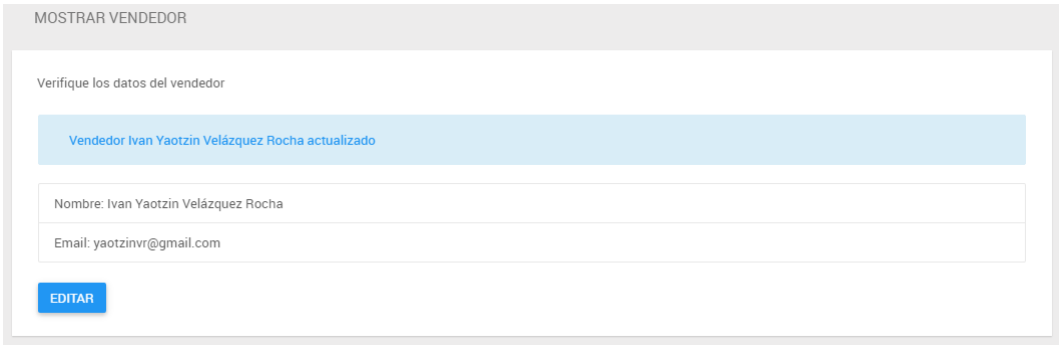


Figura 36. Detalles de vendedor actualizado.

4.2.3 Clientes

- Únicamente se puede acceder al módulo como rol Administrador y en el menú del *Home* se ve lo siguiente (Figura 37).

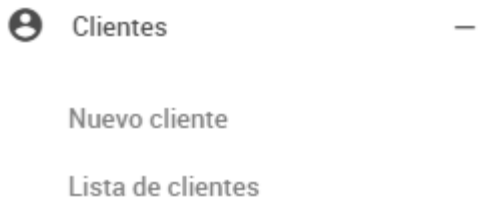


Figura 37. Menú de clientes.

- Para crear un nuevo cliente se selecciona la opción *Nuevo cliente* y se abre una nueva ventana (/tracking/cliente/create) (Figura 38).

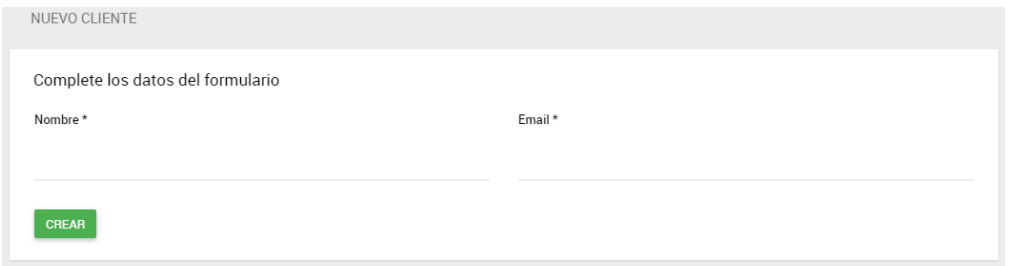


Figura 38. Nuevo cliente.

- Si se intenta crear un cliente con un email que ya exista en la base de datos el sistema arroja un error (Figura 39).

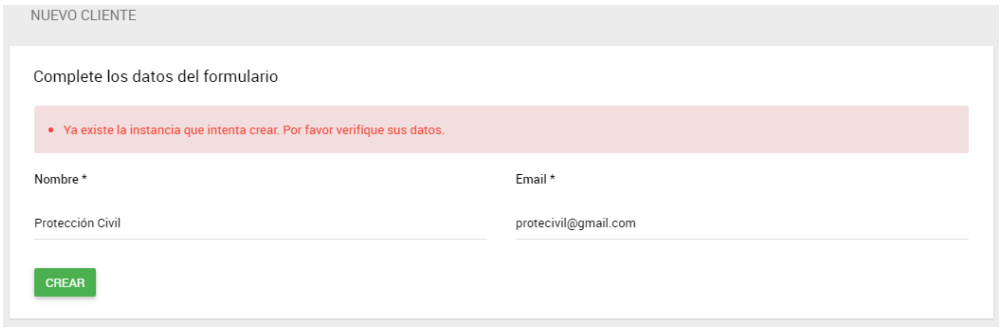


Figura 39. Error de nombre para nuevo cliente.

- Si se intenta registrar un email que no sea válido el sistema arroja un error (Figura 40).

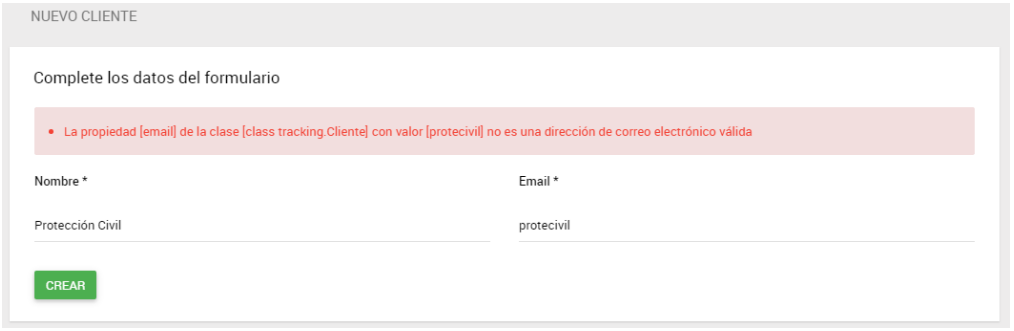


Figura 40. Error de email para nuevo cliente.

- Al crearse correctamente un cliente se muestra la siguiente ventana (Figura 41).

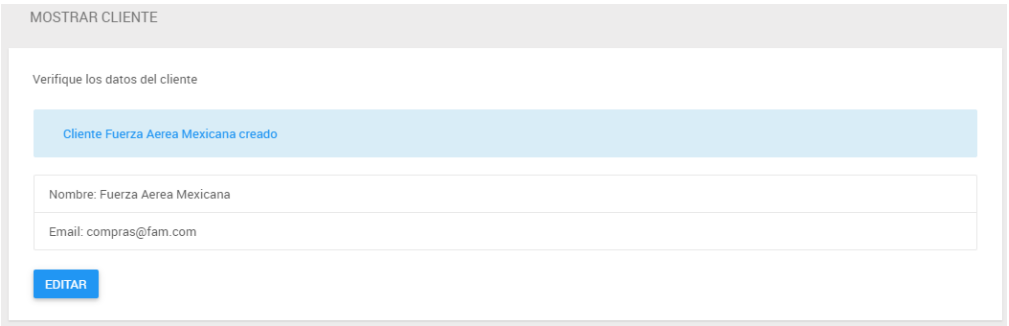


Figura 41. Creación exitosa de cliente.

- Se puede listar a todos los clientes existentes (/tracking/cliente/index) (Figura 42).

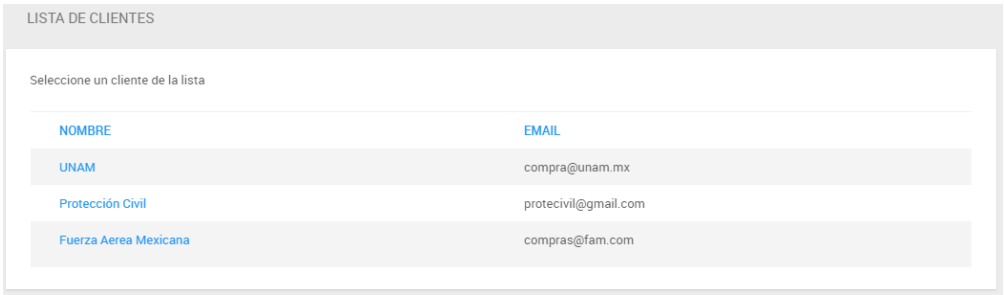


Figura 42. Lista de cliente.

- Para ver los detalles de cada uno se selecciona el nombre y se abre una nueva ventana (/tracking/cliente/show/{id}) (Figura 43).

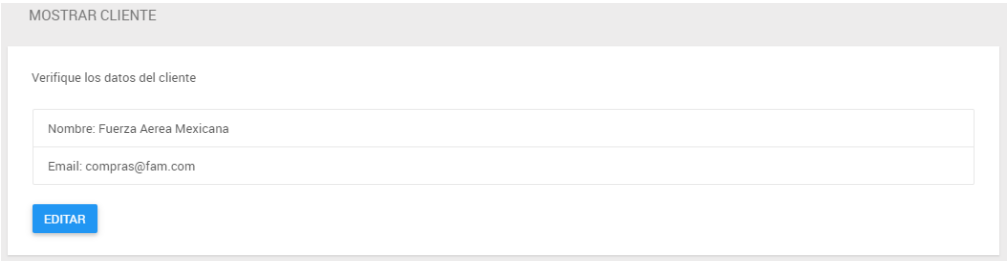


Figura 43. Detalle de cliente.

- Sólo desde esta ventana se tiene acceso a editar el cliente (/tracking/cliente/edit/{id}) (Figura 44).

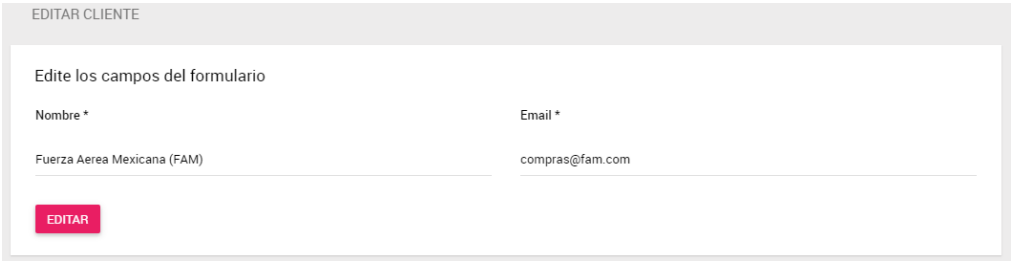


Figura 44. Editar cliente.

- Al editar se tienen las mismas reglas que al crear uno nuevo. Al actualizarlo correctamente se muestra la siguiente ventana (Figura 45).

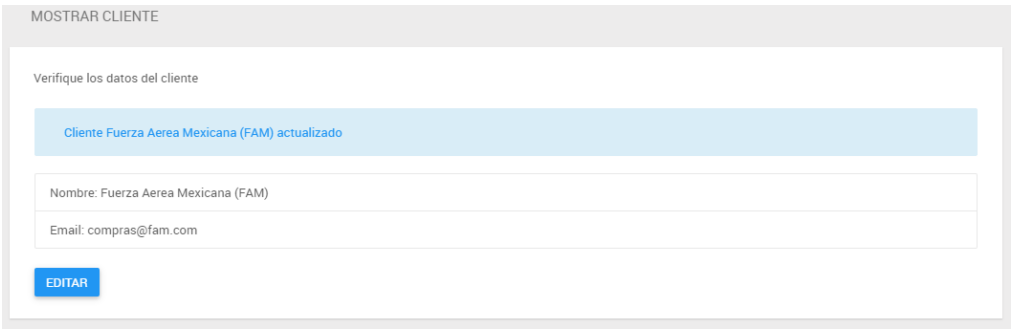


Figura 45. Detalles de cliente actualizado.

4.2.4 Proveedores

- Únicamente se puede acceder al módulo como rol Administrador y en el menú del *Home* se ve lo siguiente (Figura 46).

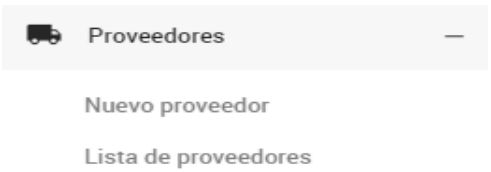


Figura 46. Menú de proveedores.

- Para crear un nuevo proveedor se selecciona la opción *Nuevo proveedor* y se abre una nueva ventana (/tracking/proveedor/create) (Figura 47).

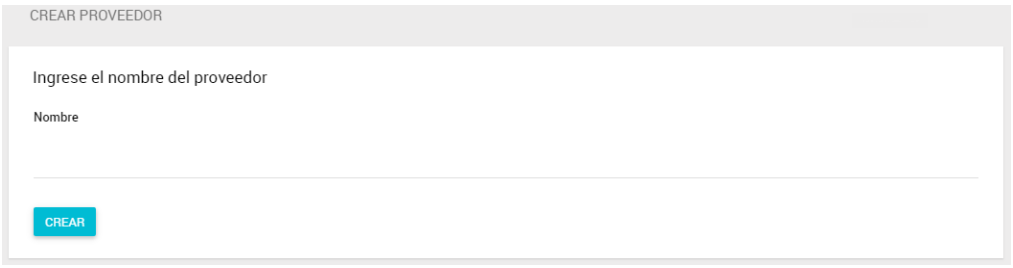


Figura 47. Crear nuevo proveedor.

- Al crearse correctamente un proveedor se muestra la siguiente ventana (Figura 48).

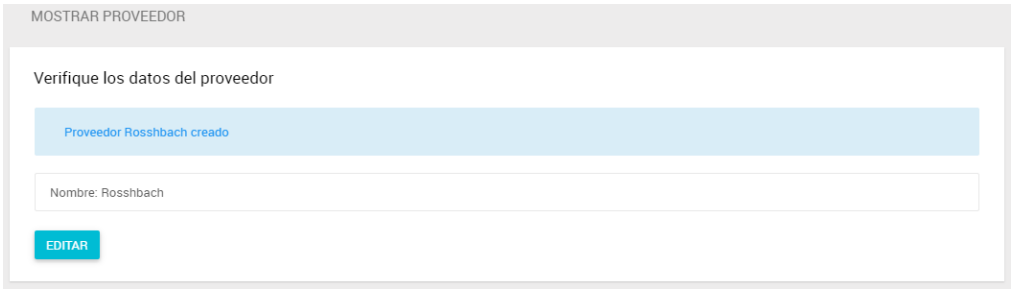


Figura 48. Proveedor creado exitosamente.

- Se puede listar a todos los proveedores existentes (/tracking/proveedor/index) (Figura 49).

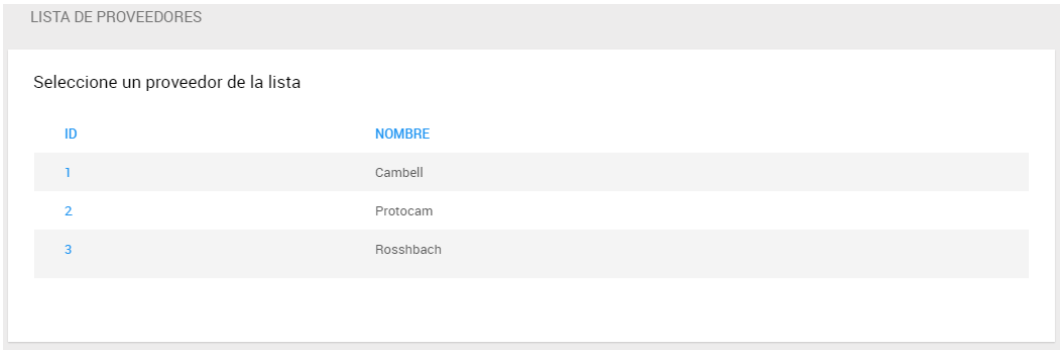


Figura 49. Lista de proveedores.

- Para ver los detalles de cada uno se selecciona el ID y se abre una nueva ventana (/tracking/proveedor/show/{id}) (Figura 50).

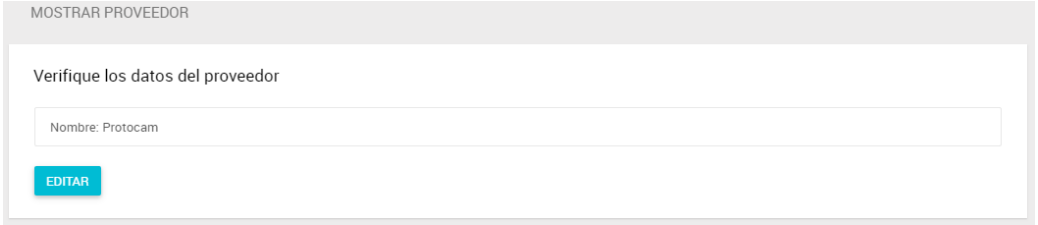


Figura 50. Detalle de proveedor.

- Sólo desde esta ventana se tiene acceso a editar el proveedor (/tracking/proveedor/edit/{id}) (Figura 51).

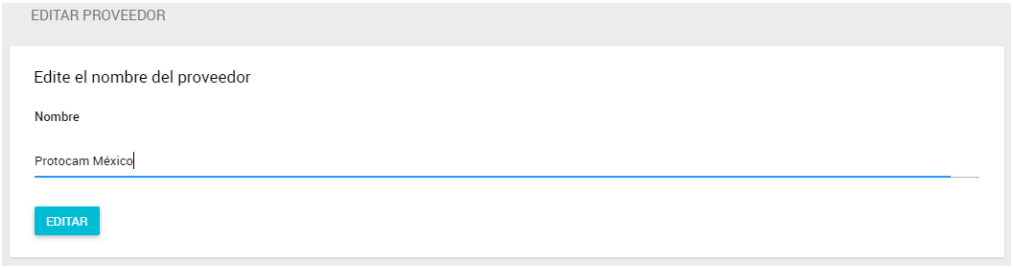


Figura 51. Editar proveedor.

- Al editar se tienen las mismas reglas que al crear uno nuevo. Al actualizarlo correctamente se muestra la siguiente ventana (Figura 52).

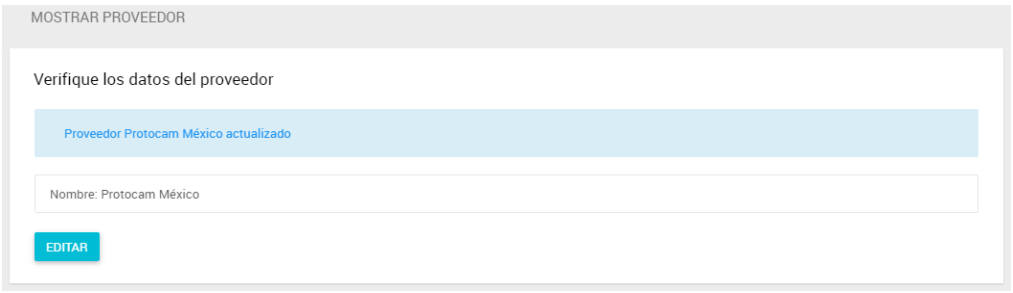


Figura 52. Detalles de proveedor actualizado.

4.2.5 Línea de Negocio

- Únicamente se puede acceder al módulo como rol Administrador y en el menú del *Home* se ve lo siguiente (Figura 53).

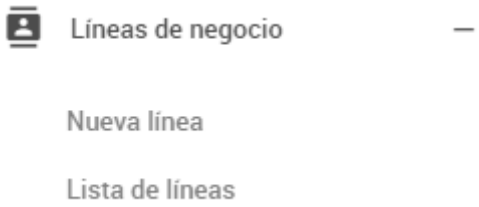


Figura 53. Menú de líneas de negocio.

- Para crear una nueva línea se selecciona la opción *Nueva línea* y se abre una nueva ventana (/tracking/lineas/create) (Figura 54).

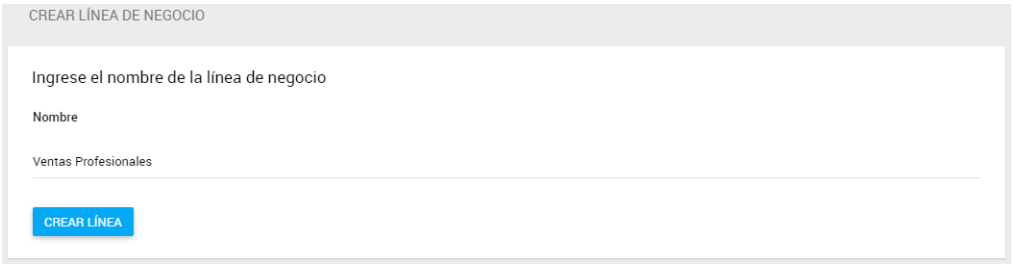


Figura 54. Crear nueva línea de negocio.

- Al crearse correctamente una línea se muestra la siguiente ventana (Figura 55).

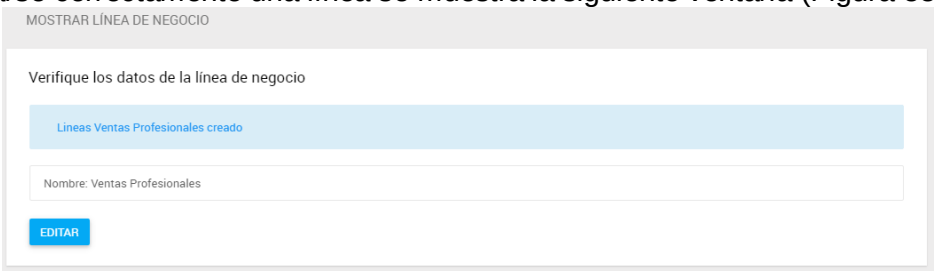


Figura 55. Nueva línea de negocio creada exitosamente.

- Se puede listar a todas las líneas existentes (/tracking/lineas/index) (Figura 56).

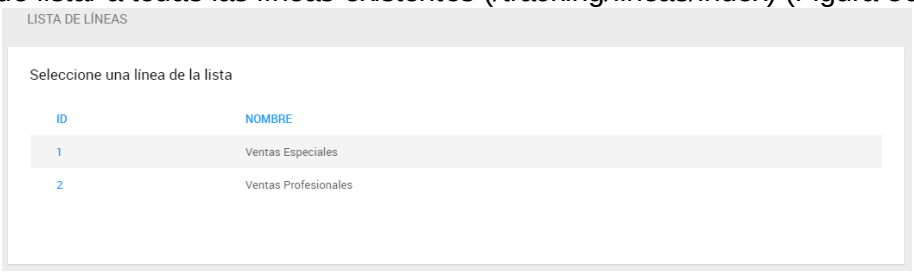


Figura 56. Lista de líneas de negocio.

- Para ver los detalles de cada una se selecciona el ID y se abre una nueva ventana (/tracking/lineas/show/{id}) (Figura 57).

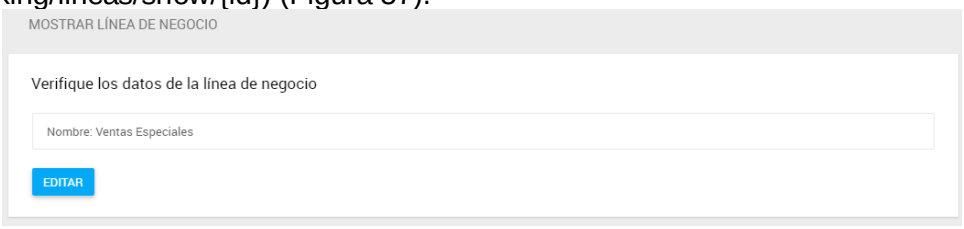


Figura 57. Detalle de línea de negocio.

- Sólo desde esta ventana se tiene acceso a editar la línea (/tracking/lineas/edit/{id}) (Figura 58).

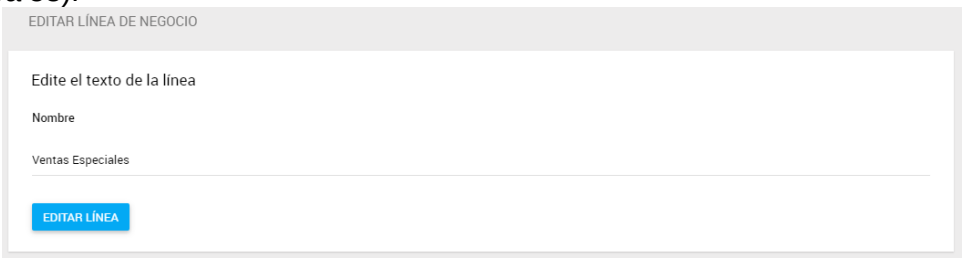


Figura 58. Editar línea de negocio.

- Al editar se tienen las mismas reglas que al crear una nueva. Al actualizarla correctamente se muestra la siguiente ventana (Figura 59).

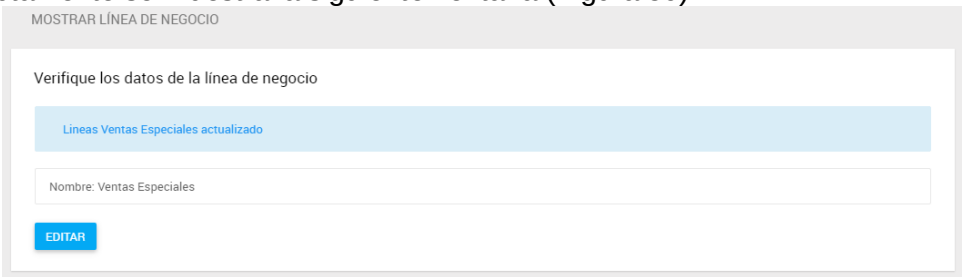


Figura 59. Detalles de línea de negocio actualizada.

4.2.6 Pedidos

- Únicamente se puede acceder al módulo como rol Administrador y en el menú del *Home* se ve lo siguiente (Figura 60).

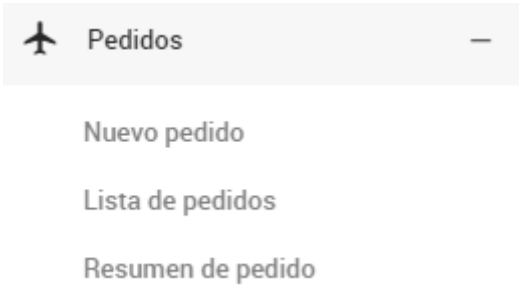


Figura 60. Menú de pedidos.

- Para crear un nuevo pedido se selecciona la opción *Nuevo pedido* y se abre una nueva ventana (/tracking/pedido/create) (Figura 61).

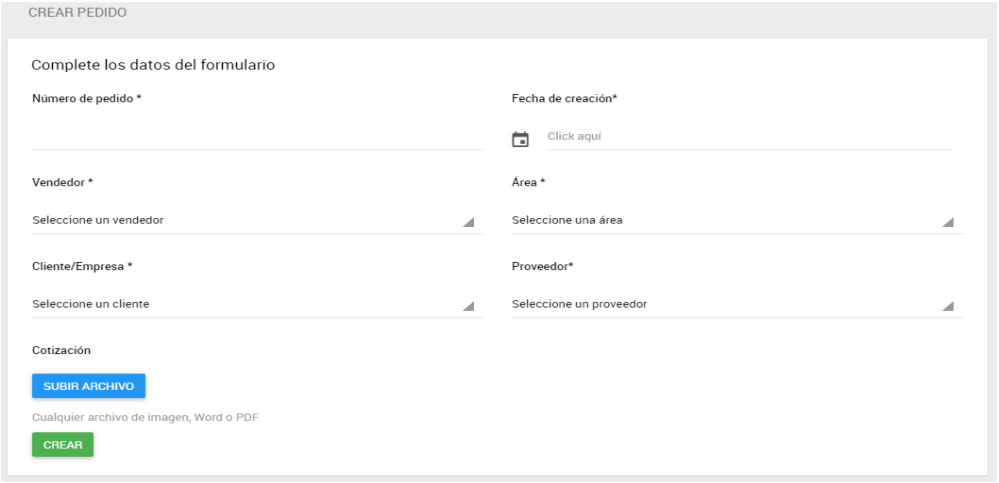


Figura 61. Crear un nuevo pedido.

- Todos los campos son obligatorios y se deben de llenar, de lo contrario no se puede crear el pedido (Figura 62).

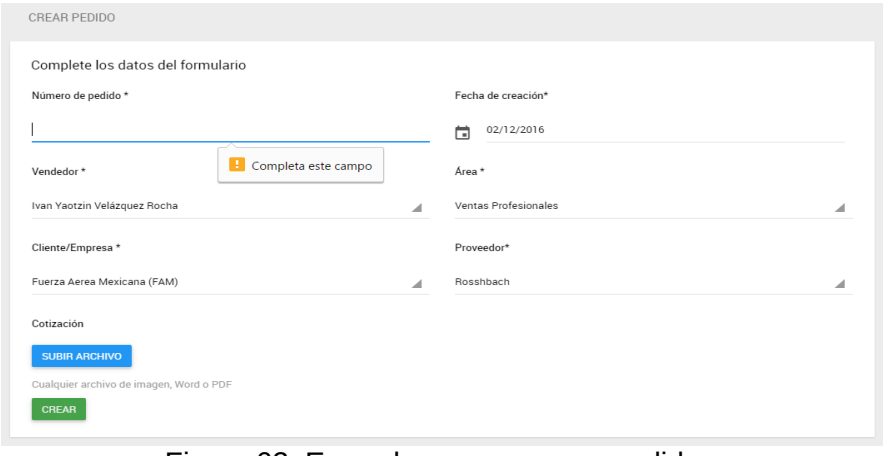


Figura 62. Error al crear un nuevo pedidor.

- Se debe de subir un archivo (imagen, documento, pdf, etc) que sea la cotización completa del pedido (Figura 63).

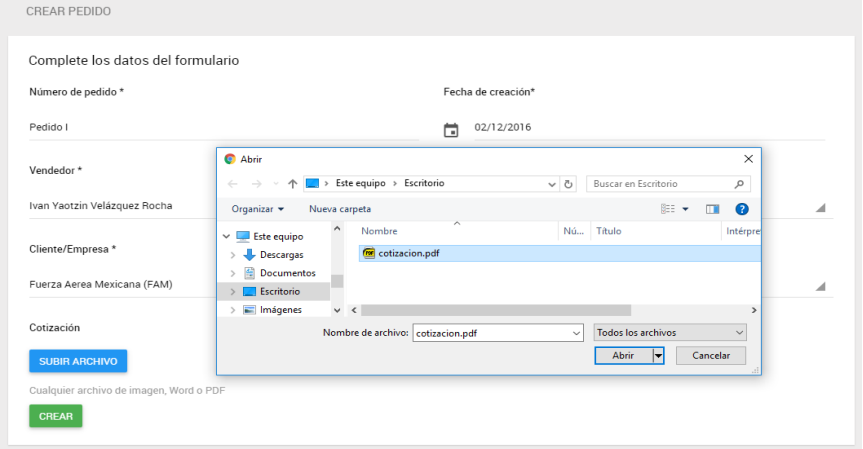


Figura 63. Adjuntar archivo al crear pedido.

- Al crearse correctamente un pedido muestras sus detalles (para descargar el archivo adjunto selecciona la opción *Ver archivo adjunto*) (Figura 64).

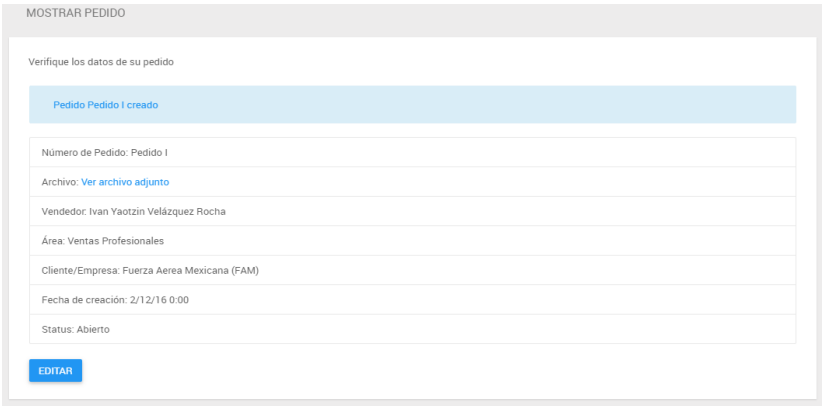


Figura 64. Nuevo pedido creado correctamente.

- Se puede listar todos los pedidos (/tracking/pedido/index) con la opción de ordenar por número de pedido, orden, vendedor, cliente o estatus; además de poder descargar directamente cualquier cotización seleccionando la opción *Ver archivo*. Cuando se crea un nuevo pedido aún no tiene orden asignada y el estatus es *Abierto* (Figura 65).

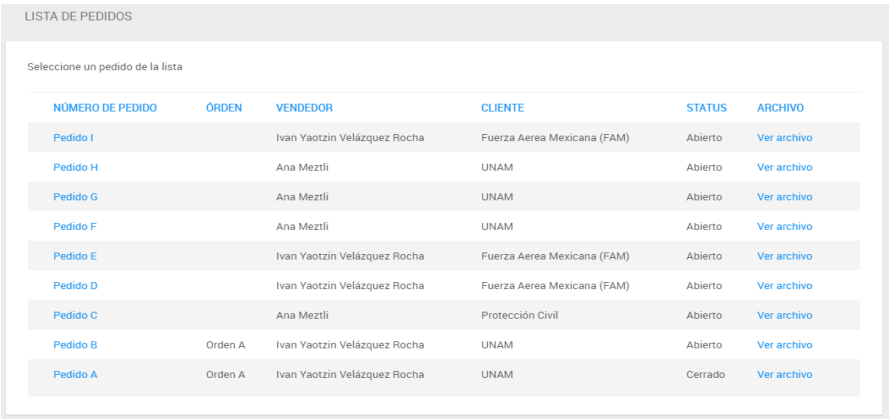


Figura 65. Lista de pedidos.

- Para ver los detalles de cada uno se selecciona el número de pedido y se abre una nueva ventana (/tracking/pedido/show/{id}) (Figura 66).

Figura 66. Detalles de pedido.

- Sólo desde esta ventana se tiene acceso a editar el pedido (/tracking/pedido/edit/{id}) (Figura 67).

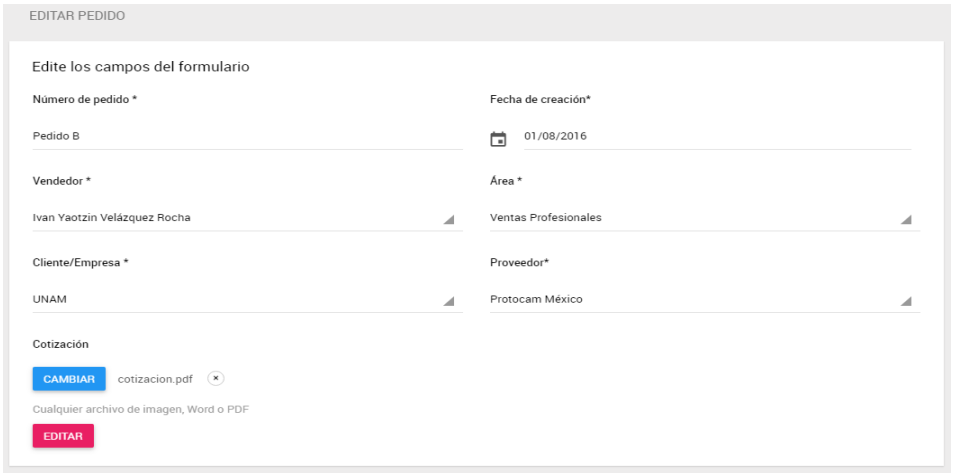


Figura 67. Editar pedido.

- Al editar se tienen las mismas reglas que al crear uno nuevo. Al actualizarlo correctamente se muestra la siguiente ventana (Figura 68).

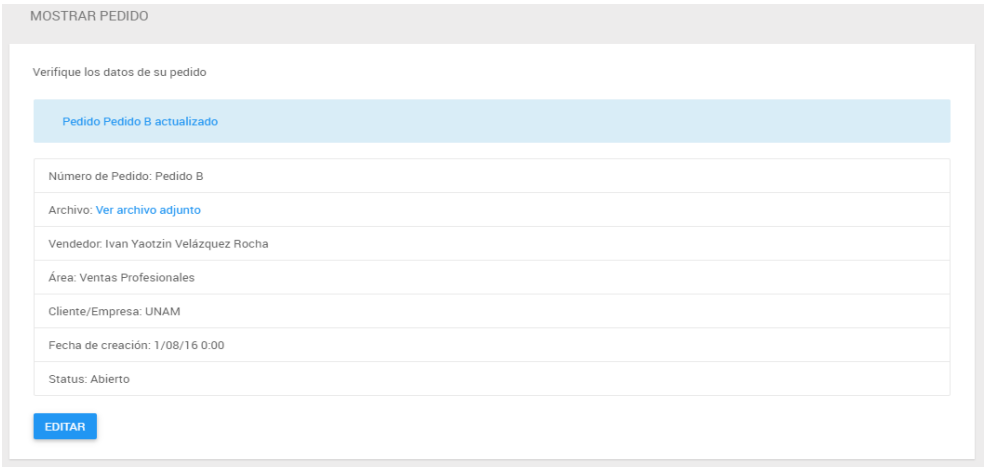


Figura 68. Detalles de pedido actualizado.

4.2.7 Tiempos

- Únicamente se puede acceder al módulo como rol Administrador y en el menú del *Home* se ve lo siguiente (Figura 69).

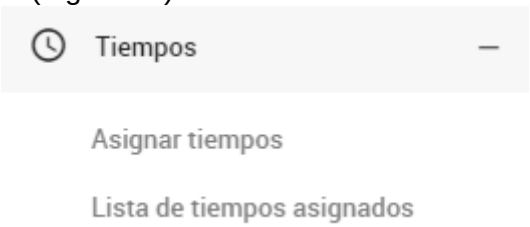


Figura 69. Menú de tiempos.

- Para crear una nueva asignación de tiempos a un proveedor se selecciona la opción *Asignar tiempos* y se abre una nueva ventana (/tracking/tiempos/create) (Figura 70).

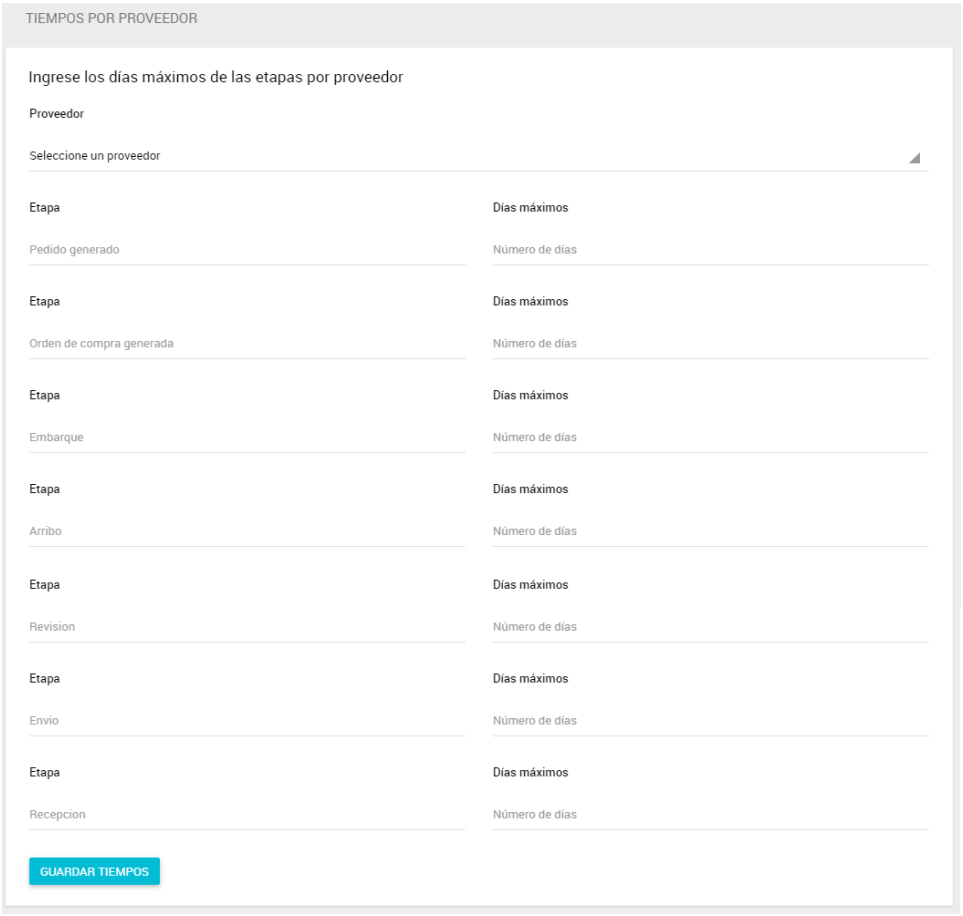


Figura 70. Creación de tiempos.

- Todos los campos son obligatorios y se deben de llenar, de lo contrario no te deja asignar los tiempos. Se debe de elegir un proveedor de la lista, y sólo mostrará aquellos que aún no tengan tiempos asignados (Figura 71).

Figura 71. Elegir proveedor para asignar tiempos.

- Si no se ha elegido un proveedor se manda el siguiente aviso y no se puede avanzar (Figura 72).

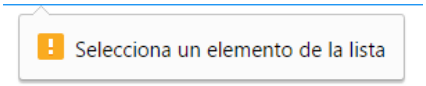


Figura 72. Error al crear nuevos tiempos por no elegir proveedor.

- Por cada una de las siete etapas se debe de agregar un máximo de días que puede tardar el pedido en esa etapa (Figura 73).

Figura 73. Agregar tiempo de atención por etapa.

- Si no se ha agregado días máximos a alguna etapa se manda el siguiente aviso y no se puede avanzar (Figura 74).

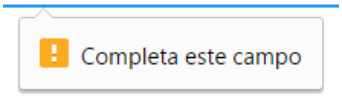


Figura 74. Error por no agregar tiempo de atención por etapa.

- Al llenarse correctamente el formulario y seleccionar la opción *Guardar Tiempos* se muestra los detalles de la asignación de tiempos (Figura 75).

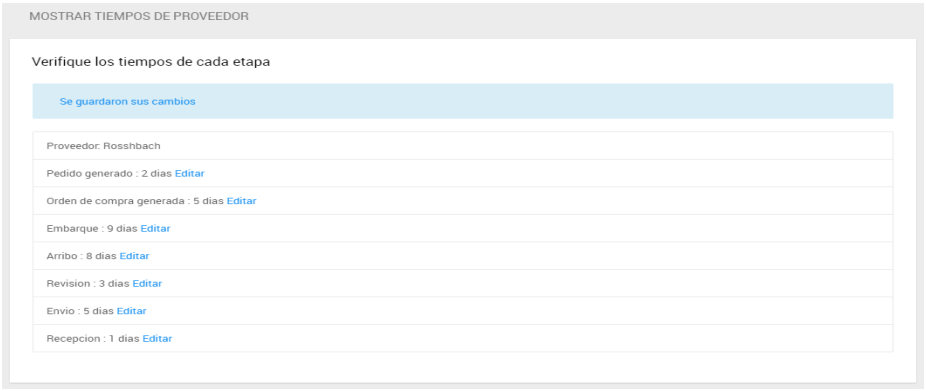


Figura 75. Nuevos tiempos creados correctamente.

- Se puede listar todos los tiempos existentes (/tracking/tiempos/index) donde se puede ver la relación entre proveedor, etapas (creación, orden de compra, embarque, arribo, revisión, envío, recepción) y total de tiempo máximo que puede tardar cierto proveedor en surtir una orden. Sólo se muestran aquí los proveedores que ya tienen tiempos asignados (Figura 76).

Figura 76. Lista de tiempos.

- Para ver los detalles de cada uno se debe de dar click sobre el proveedor y se abre una nueva ventana (/tracking/tiempos/show/{id}) (Figura 77).

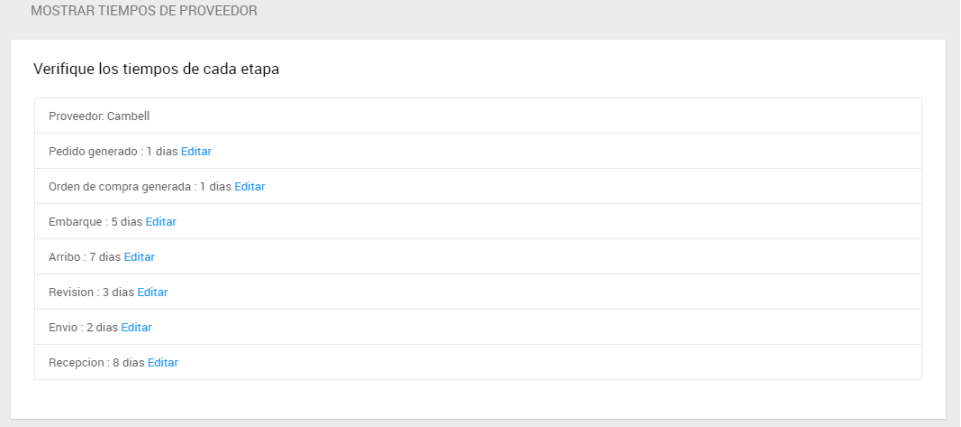


Figura 77. Detalle de tiempos.

- Sólo desde esta ventana se tiene acceso a editar algún tiempo del proveedor (/tracking/tiempos/edit/{id}). Para esto se elige la etapa de la cual editar el tiempo máximo que puede tardar y se selecciona la opción *Editar* (Figura 78).

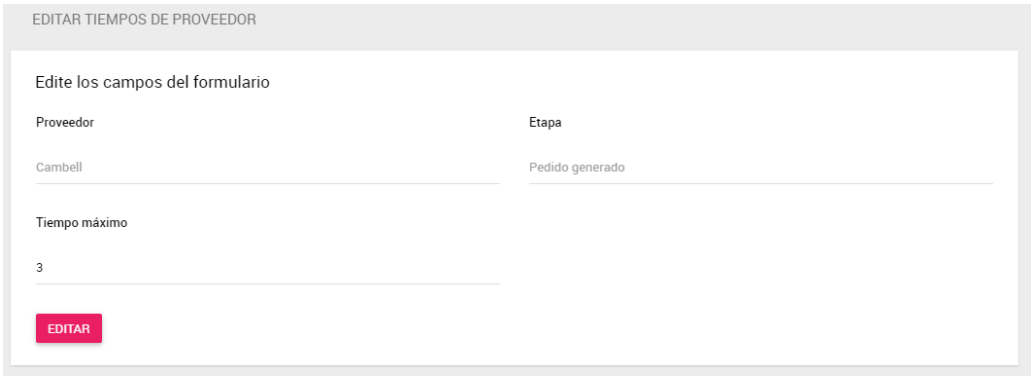


Figura 78. Editar tiempos.

- Al editar se tienen las mismas reglas que al crear uno nuevo (todos los campos son obligatorios). Al actualizarlo correctamente se muestran los detalles nuevamente (Figura 79).

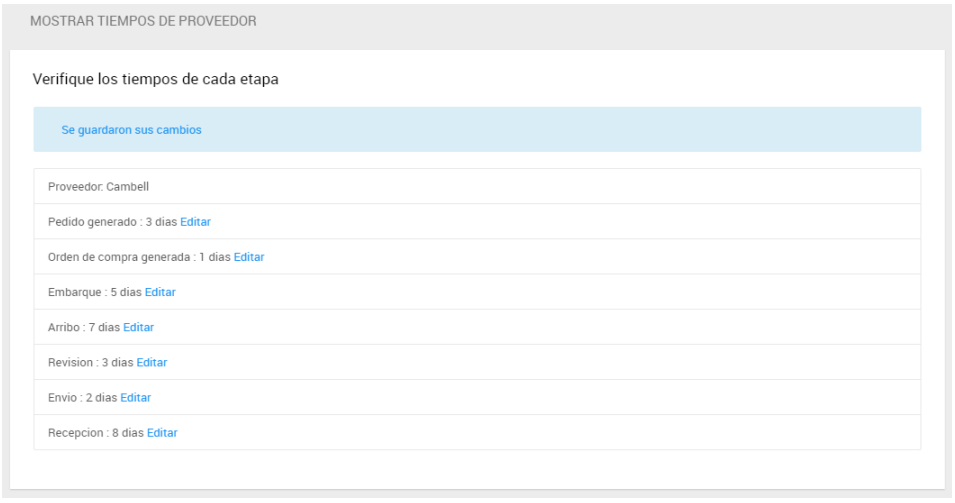


Figura 79. Detalle de tiempo actualizado.

4.2.8 Administración

Orden de compra:

- Únicamente se puede acceder al módulo como rol Administrador y en el menú del *Home* se ve lo siguiente (Figura 80).

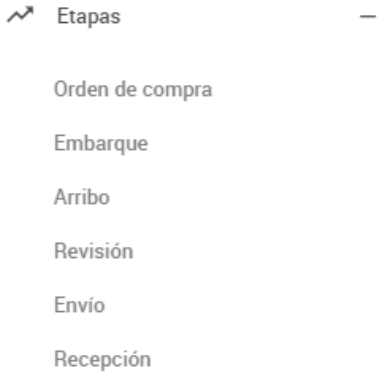


Figura 80. Menú de etapas.

- Para generar una nueva orden de compra se selecciona la opción *Orden de compra* y se abre una nueva ventana (/tracking/tracking/ordenDeCompra) (Figura 81).

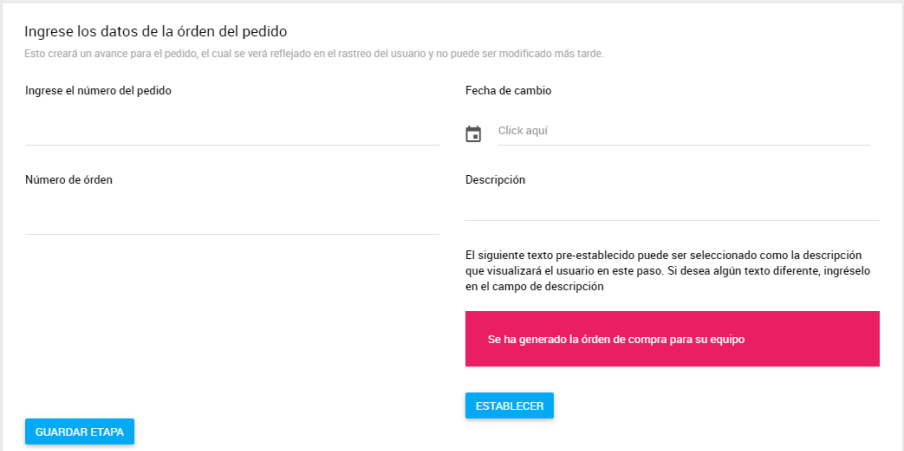


Figura 81. Crear una nueva orden de compra.

- Todos los campos son obligatorios y se deben de llenar, de lo contrario no deja crear la orden de compra del pedido. Se debe de ingresar la fecha del cambio de etapa del pedido, una descripción (selecciona la opción *Establecer* para generar una descripción por defecto), el número de pedido que se va a asociar a la orden de compra y el número de orden de compra (se pueden asociar más de un pedido a una orden).
- Debe de existir el pedido, de lo contrario no dejará crear el avance y se genera el siguiente aviso (Figura 82).

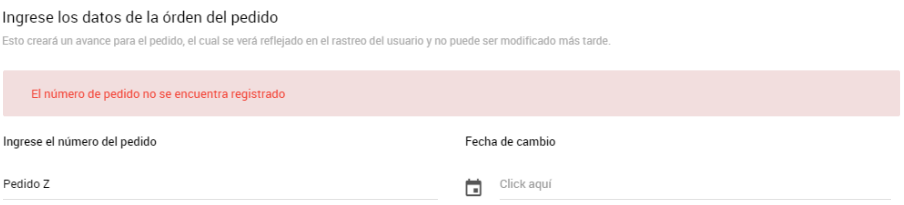


Figura 82. Número de pedidor no existe para nueva orden de compra.

- Si el pedido ya ha sido procesado para esta etapa no dejará crear el avance y se genera el siguiente aviso (Figura 83).

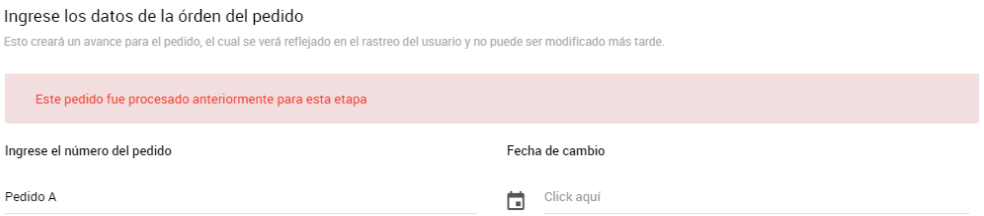


Figura 83. Número de pedidor ya procesado para nueva orden de compra.

- Cuando se crea correctamente el avance, se redirige al *Home*, y en la tabla de *Movimientos Recientes* se puede ver el nuevo avance generado (Figura 84).

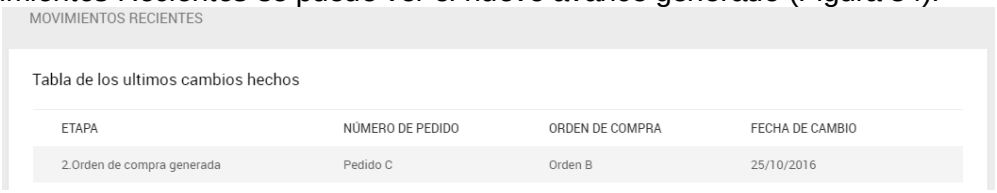


Figura 84. Nueva orden de compra creada.

Embarque:

- Para generar un nuevo avance de embarque de un pedido se selecciona la opción *Embarque* del menú de *Home* y se abre una nueva ventana (/tracking/tracking/embarque) (Figura 85).

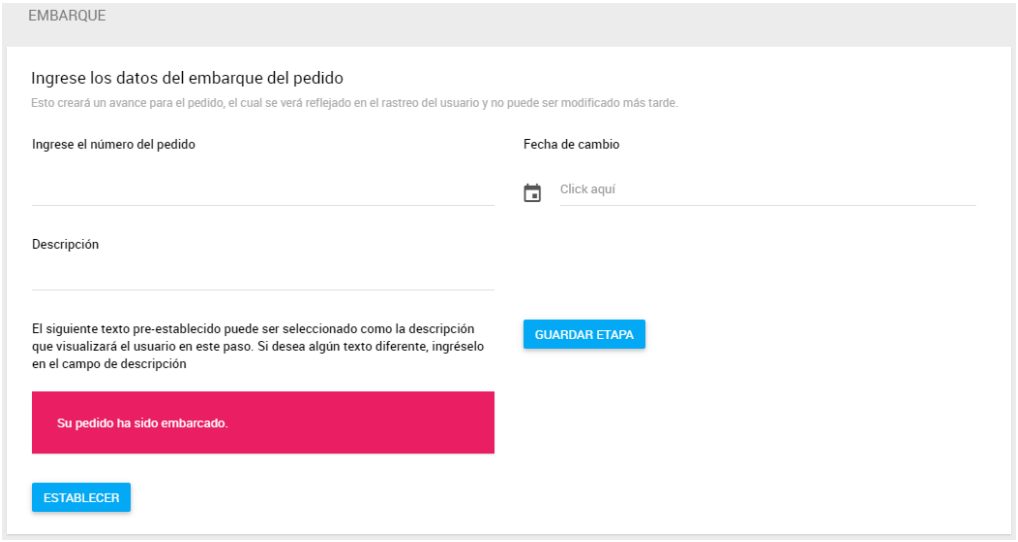


Figura 85. Crear nuevo embarque.

- Todos los campos son obligatorios y se deben de llenar, de lo contrario no deja crear el embarque del pedido. Se debe de ingresar la fecha del cambio de etapa del pedido, una descripción (selecciona la opción Establecer para generar una descripción por defecto) y el número de pedido que está siendo enviado por el proveedor.
- Debe de existir el pedido, de lo contrario no dejará crear el avance y se genera el siguiente aviso (Figura 86).

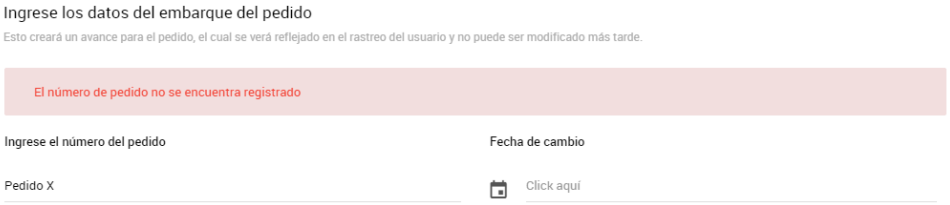


Figura 86. Número de pedido no existe para nuevo embarque.

- Si el pedido ya ha sido procesado para esta etapa no dejará crear el avance y se genera el siguiente aviso (Figura 87).

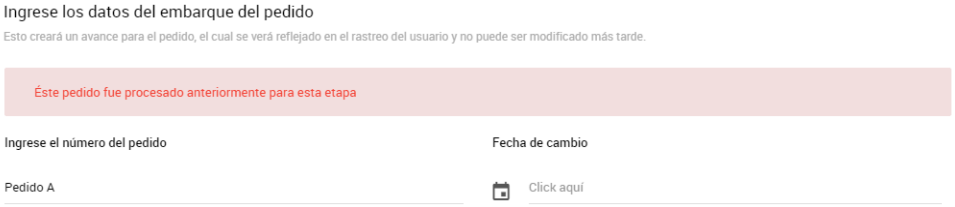


Figura 87. Número de pedido ya procesado para nuevo embarque.

- Si el pedido no está procesado justo en la etapa anterior no dejará crear el avance y se genera el siguiente aviso (Figura 88).

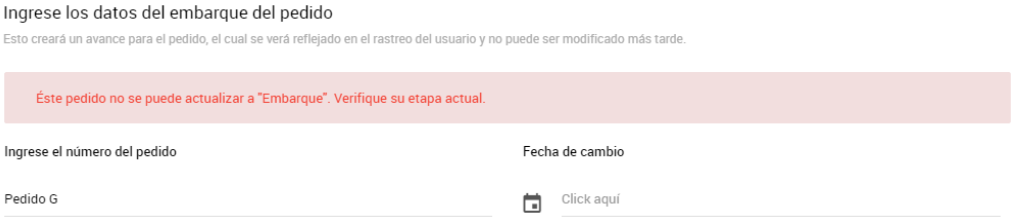


Figura 88. Número de pedido no disponible para nuevo embarque.

- Cuando se crea correctamente el avance, se redirige al *Home*, y en la tabla de *Movimientos Recientes* se puede ver el nuevo avance generado (Figura 89).

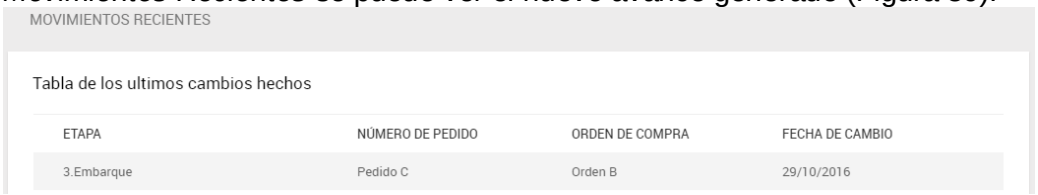


Figura 89. Embarque creado.

Arribo:

- Para generar un nuevo avance de arribo de un pedido se selecciona la opción *Arribo* del menú de *Home* y se abre una ventana (/tracking/tracking/arribo) (Figura 90).

Figura 90. Crear nuevo arribo.

- Todos los campos son obligatorios y se deben de llenar, de lo contrario no deja crear el arribo del pedido. Se debe de ingresar la fecha del cambio de etapa del pedido, una descripción (selecciona la opción *Establecer* para generar una descripción por defecto) y el número de pedido que ha llegado a la empresa.
- Debe de existir el pedido, de lo contrario no dejará crear el avance y se genera el siguiente aviso (Figura 91).

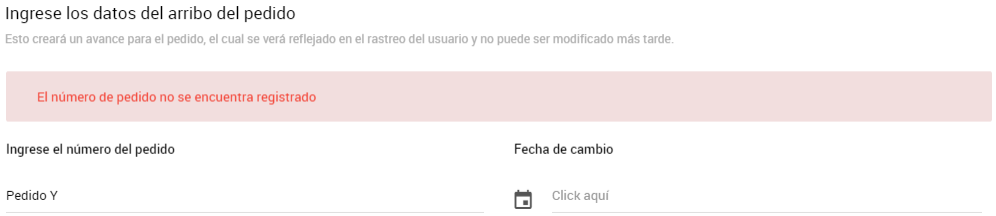


Figura 91. Número de pedido no existe nuevo arribo.

- Si el pedido ya ha sido procesado para esta etapa no dejará crear el avance y se genera el siguiente aviso (Figura 92).

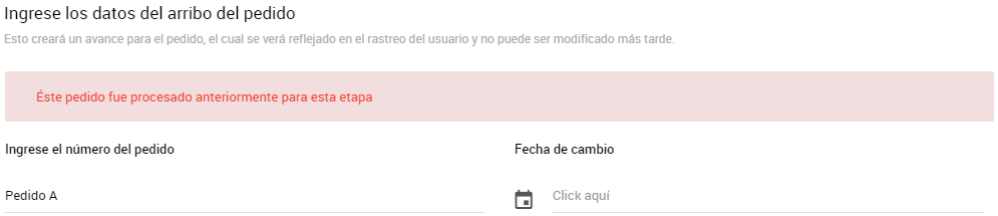


Figura 92. Número de pedido ya procesado para nuevo arribo.

- Si el pedido no está procesado justo en la etapa anterior no dejará crear el avance y se genera el siguiente aviso (Figura 93).

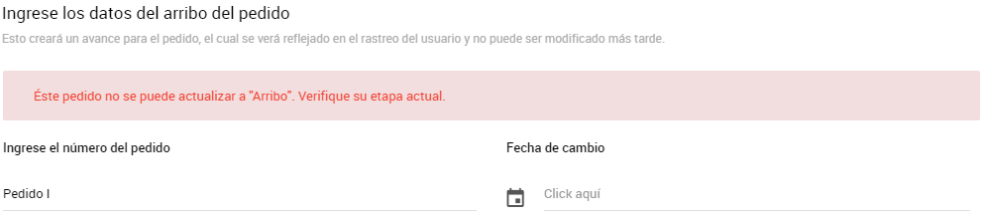


Figura 93. Número de pedido no disponible para nuevo arribo.

- Cuando se crea correctamente el avance, se redirige al *Home*, y en la tabla de *Movimientos Recientes* se puede ver el nuevo avance generado (Figura 94).

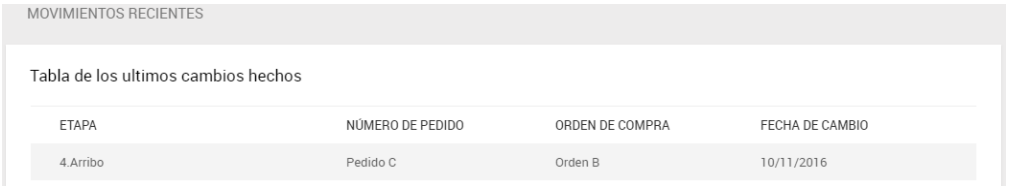


Figura 94. Nuevo arribo creado.

Revisión:

- Para generar un nuevo avance de revisión de un pedido se selecciona la opción *Revisión* del menú de *Home* y se abre una nueva ventana (/tracking/tracking/arribo) (Figura 95).

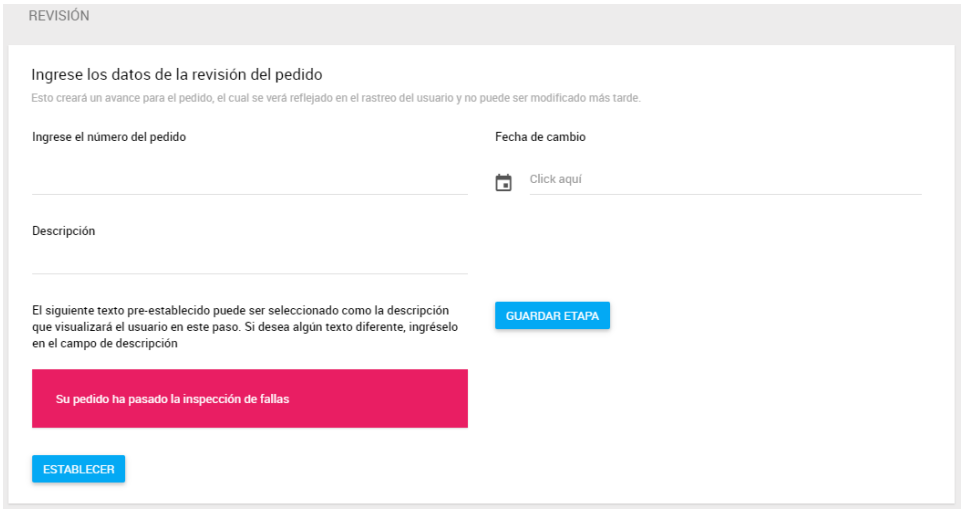


Figura 95. Crear nueva revisión.

- Todos los campos son obligatorios y se deben de llenar, de lo contrario no deja crear la revisión del pedido. Se debe de ingresar la fecha del cambio de etapa del pedido, una descripción (selecciona la opción *Establecer* para generar una descripción por defecto) y el número de pedido que ha de revisar.
- Debe de existir el pedido, de lo contrario no dejará crear el avance y se genera el siguiente aviso (Figura 96).

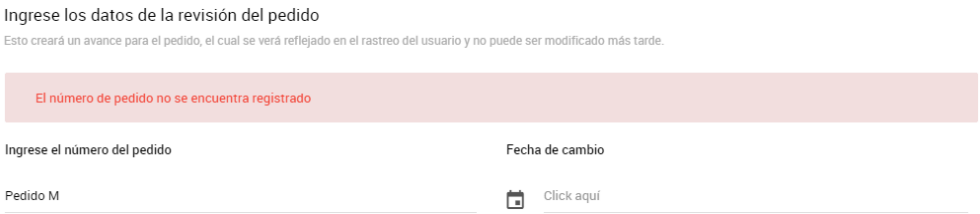


Figura 96. Número de pedido no existe para nueva revisión.

- Si el pedido ya ha sido procesado para esta etapa no dejará crear el avance y se genera el siguiente aviso (Figura 97).

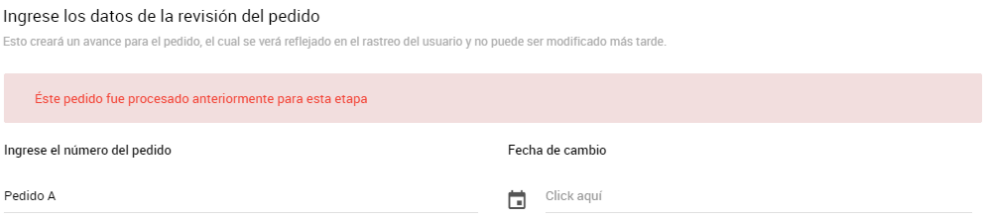


Figura 97. Número de pedido ya procesado para nueva revisión.

- Si el pedido no está procesado justo en la etapa anterior no dejará crear el avance y se genera el siguiente aviso (Figura 98).

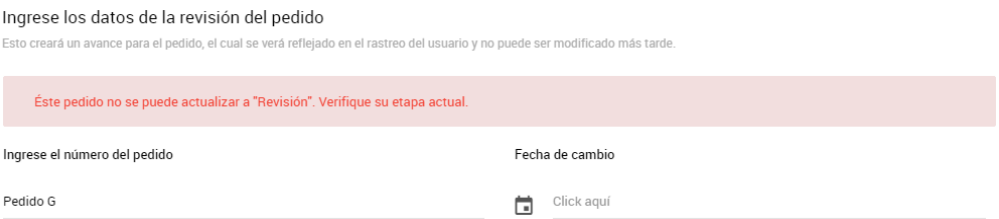


Figura 98. Número de pedido no disponible para nueva revisión.

- Cuando se crea correctamente el avance, se redirige al *Home*, y en la tabla de *Movimientos Recientes* se puede ver el nuevo avance generado (Figura 99).

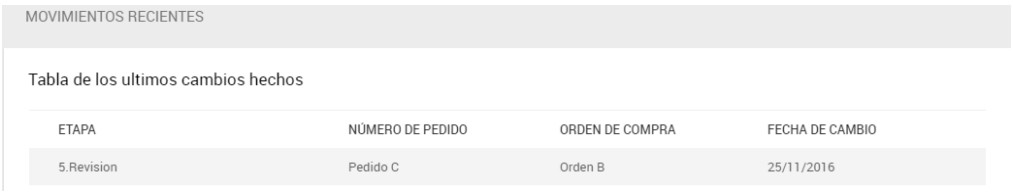


Figura 99. Nueva revisión creada.

Envío:

- Para generar un nuevo avance de envío de un pedido se selecciona la opción *Envío* del menú de *Home* y se abre una nueva ventana (/tracking/tracking/envio) (Figura 100).

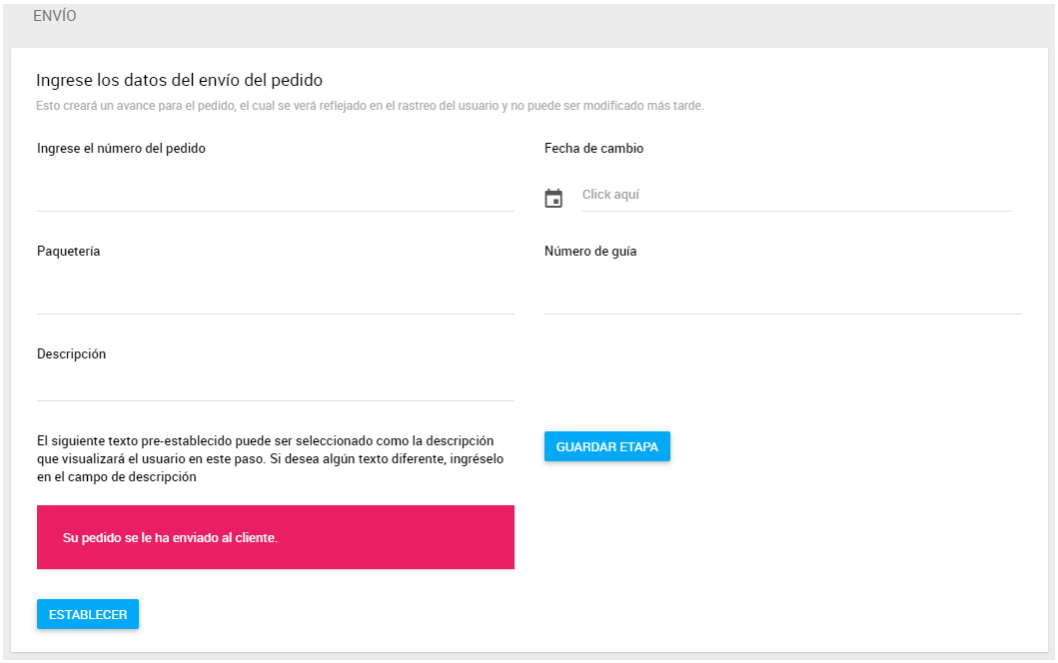


Figura 100. Crear nuevo envío.

- Todos los campos son obligatorios y se deben de llenar, de lo contrario no deja crear el envío del pedido. Se debe de ingresar la fecha del cambio de etapa del pedido, una descripción (selecciona la opción *Establecer* para generar una descripción por defecto) y el número de pedido que se le va a enviar al cliente.
- Debe de existir el pedido, de lo contrario no dejará crear el avance y se genera el siguiente aviso (Figura 101).

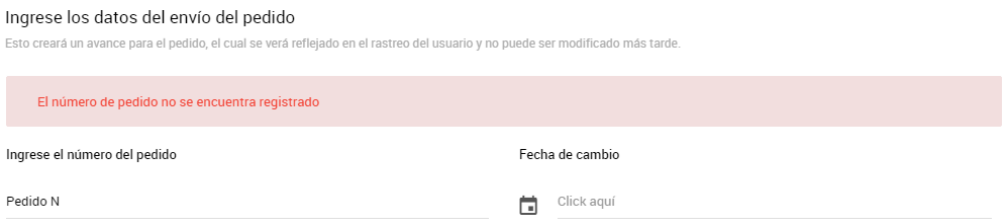


Figura 101. Número de pedido no existe para nuevo envío.

- Si el pedido ya ha sido procesado para esta etapa no dejará crear el avance y se genera el siguiente aviso (Figura 102).

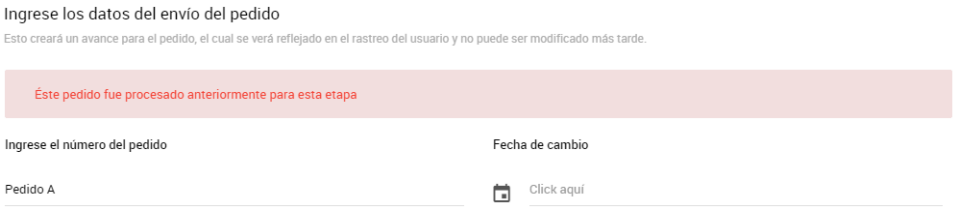


Figura 102. Número de pedido ya procesado para nuevo envío.

- Si el pedido no está procesado justo en la etapa anterior no dejará crear el avance y se genera el siguiente aviso (Figura 103).

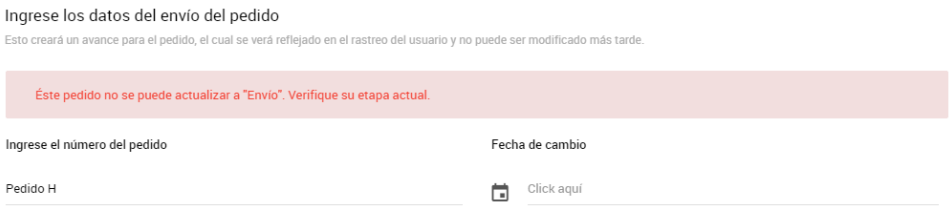


Figura 103. Número de pedido no disponible para nuevo envío.

- Cuando se crea correctamente el avance, se redirige al *Home*, y en la tabla de *Movimientos Recientes* se puede ver el nuevo avance generado (Figura 104).

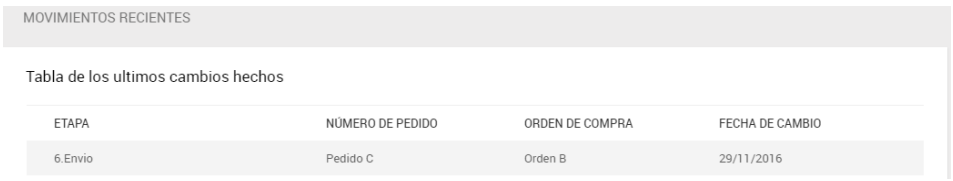


Figura 104. Nuevo envío creado.

Recepción:

- Para generar un nuevo avance de recepción de un pedido se selecciona la opción *Recepción* del menú de *Home* y abre una nueva ventana (/tracking/tracking/recepcion) (Figura 105).

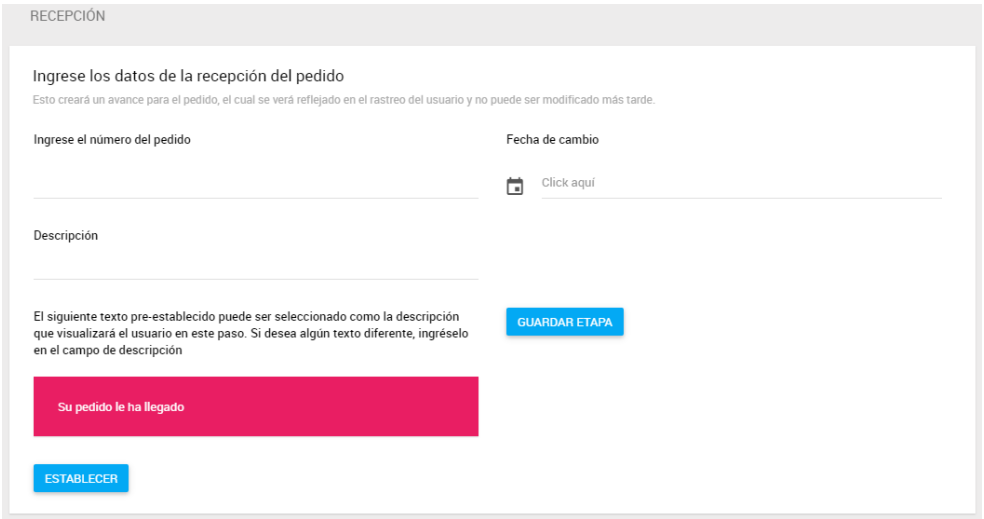


Figura 105. Crear nueva recepción.

- Todos los campos son obligatorios y se deben de llenar, de lo contrario no deja crear la recepción del pedido. Se debe de ingresar la fecha del cambio de etapa del pedido, una descripción (selecciona la opción *Establecer* para generar una descripción por defecto) y el número de pedido que se ha recibido el cliente.
- Debe de existir el pedido, de lo contrario no dejará crear el avance y se genera el siguiente aviso (Figura 106).

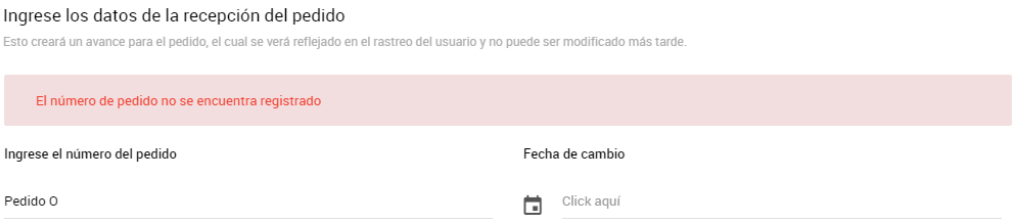


Figura 106. Número de pedido no existe para nueva recepción.

- Si el pedido ya ha sido procesado para esta etapa no dejará crear el avance y se genera el siguiente aviso (Figura 107).

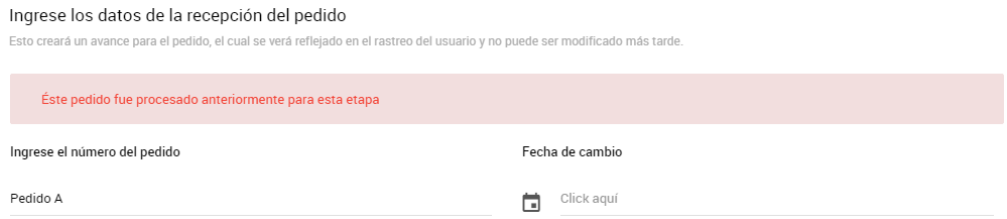


Figura 107. Número de pedido ya procesado para nueva recepción.

- Si el pedido no está procesado justo en la etapa anterior no dejará crear el avance y se genera el siguiente aviso (Figura 108).

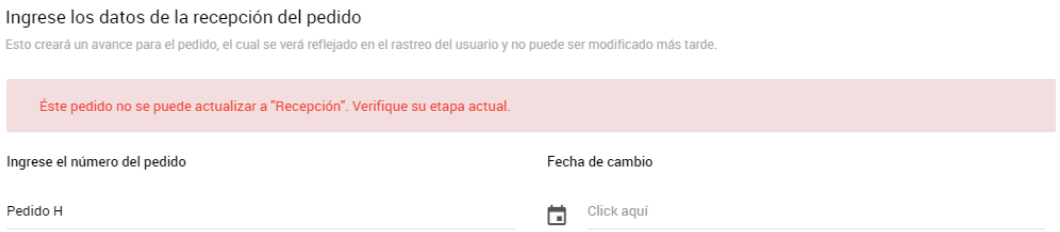


Figura 108. Número de pedido no disponible para nueva recepción.

- Cuando se crea correctamente el avance, se redirige al *Home*, y en la tabla de *Movimientos Recientes* se puede ver el nuevo avance generado. Además, el estatus del pedido se actualiza a *Cerrado* (Figura 109).

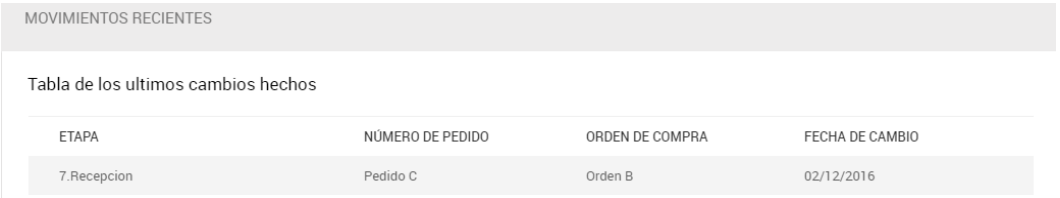


Figura 109. Nueva recepción creada.

4.2.9 Estadísticas

De manera predeterminada, al entrar a la aplicación, la página inicial es *Home* (/tracking) y en cualquier momento se puede regresar al *Home* en la primera opción del menú principal (Figura 110).

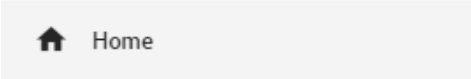


Figura 110. Menú para estadísticas.

Tanto el rol de Administrador como de Usuario pueden acceder a este módulo. Se puede ver la siguiente información.

- **Pedidos por mes**: esta gráfica muestra cuántos pedidos ha habido por mes durante el año en curso. En el eje horizontal muestra los meses y en el vertical el número de pedidos (Figura 111).

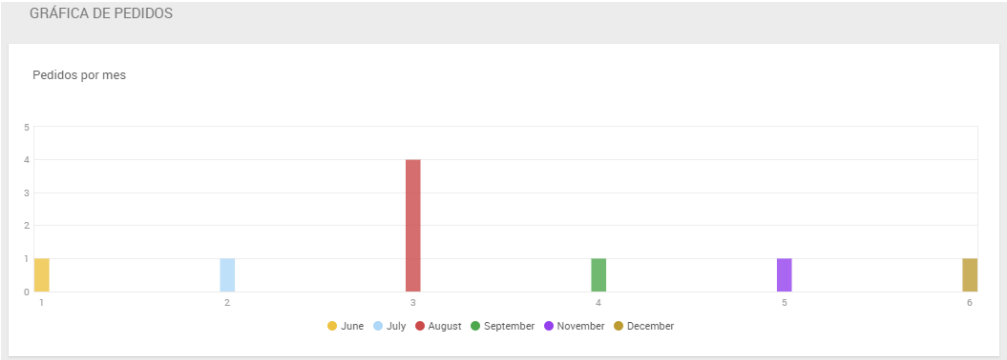
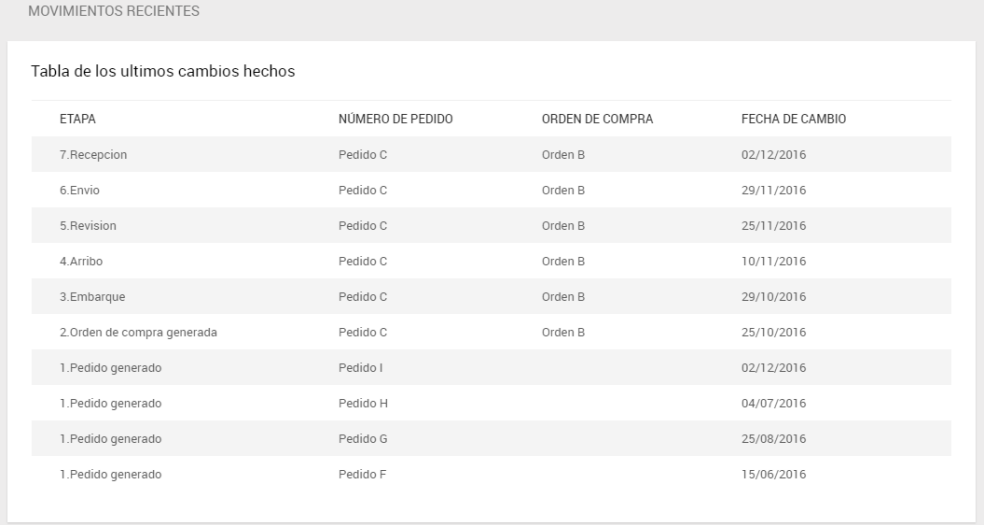


Figura 111. Gráfica de pedidos por mes.

- **Movimientos recientes**: esta es una tabla que muestra, en orden de fecha, los últimos 10 cambios de etapas que se han hecho a los pedidos (Figura 112).



ETAPA	NÚMERO DE PEDIDO	ORDEN DE COMPRA	FECHA DE CAMBIO
7.Recepcion	Pedido C	Orden B	02/12/2016
6.Envio	Pedido C	Orden B	29/11/2016
5.Revision	Pedido C	Orden B	25/11/2016
4.Arribo	Pedido C	Orden B	10/11/2016
3.Embarque	Pedido C	Orden B	29/10/2016
2.Orden de compra generada	Pedido C	Orden B	25/10/2016
1.Pedido generado	Pedido I		02/12/2016
1.Pedido generado	Pedido H		04/07/2016
1.Pedido generado	Pedido G		25/08/2016
1.Pedido generado	Pedido F		15/06/2016

Figura 112. Tabla de movimientos recientes.

- **Pedidos por etapa**: esta es una sección donde se puede visualizar el número de pedidos que se encuentran en cada etapa (Figura 113).

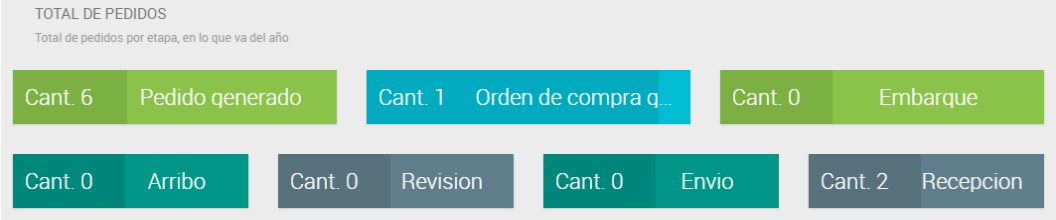


Figura 113. Número de pedidos por etapa.

- Si se selecciona una etapa se abrirá una nueva ventana (/tracking/home/pedidosEtapa/{id}) donde se muestra a detalle los pedidos que se encuentran en la etapa seleccionada. Las columnas son: número de pedido, orden, vendedor, cliente, status y archivo (se puede descargar el archivo adjunto si se selecciona la opción *Ver archivo*) (Figura 114).

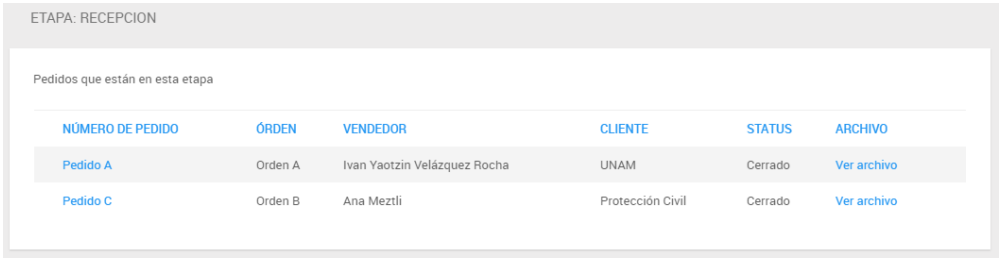


Figura 114. Pedidos de cierta etapa.

- **Ventas por departamento**: este es un apartado donde se muestran el total de pedidos que cada departamento ha vendido en lo que va del año (Figura 115).

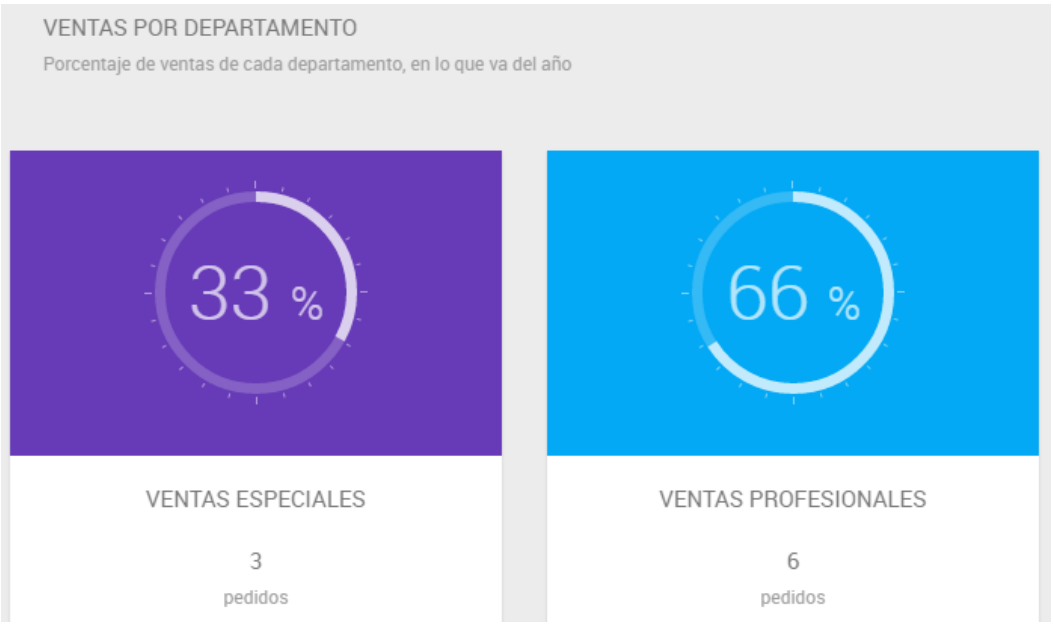


Figura 115. Gráfica de ventas por departamento.

- **Pedidos por cliente**: aquí se muestra una gráfica de pastel con los 10 mejores clientes y el porcentaje de pedidos que han hecho, en lo que va del año. Además, una tabla donde dice por cada cliente (de los 10 mejores) el número de pedidos que ha hecho, en lo que va del año (Figura 116).

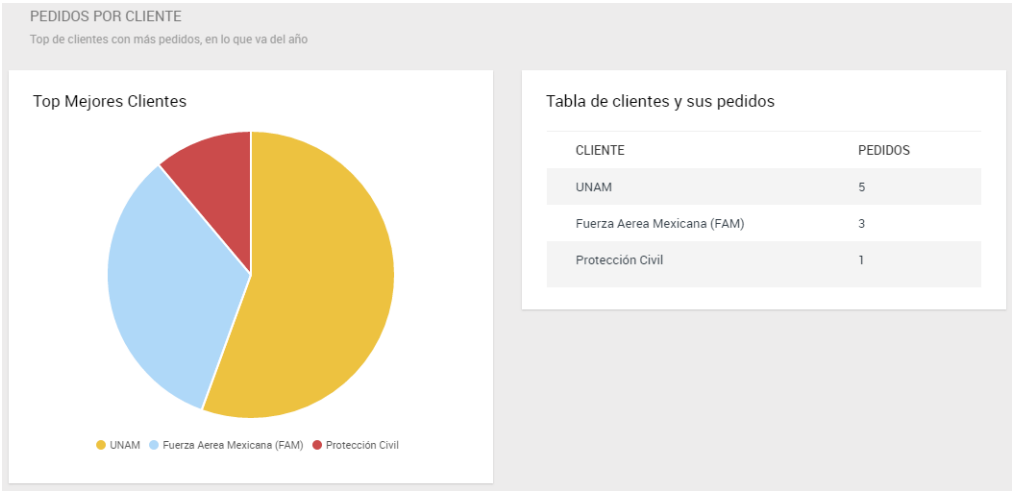


Figura 116. Gráfica de pedidos por cliente.

- Si se selecciona el nombre de un cliente se abre una nueva ventana (/tracking/home/pedidosCliente/{id}) donde se muestra a detalle los pedidos que ha hecho el cliente seleccionado. Las columnas son: número de pedido, orden, vendedor, cliente, status y archivo (se puede descargar el archivo adjunto si se selecciona la opción *Ver archivo*) (Figura 117).

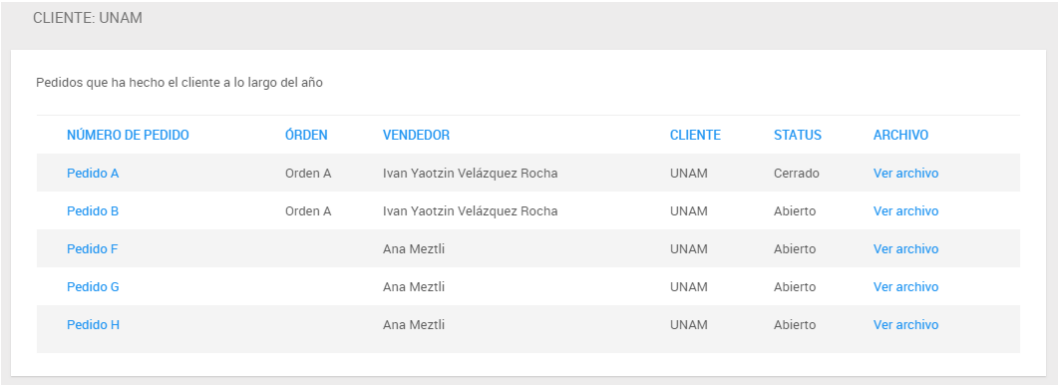


Figura 117. Pedidos de cierto cliente.

Capítulo 5. Conclusiones

A lo largo del presente trabajo mostré de forma detallada la manera en que se implementó el sistema TRACKING. Mostré de manera integral todas las partes que componen al desarrollo de software desde el punto de vista de la ingeniería de software. Cumpliendo así el principal objetivo del presente documento.

El segundo objetivo también se cumplió satisfactoriamente, dado que hubo todo un capítulo destinado a mostrar el funcionamiento de la aplicación que se desarrolló: *Producción*. Este capítulo se divide en dos subcapítulos: *Puesta en Producción* y *Flujo de Trabajo*. En este segundo subcapítulo se explicó a detalle cada módulo del sistema, su interacción con el usuario y sus restricciones; por medio de ejemplos sencillos. Aquí es donde queda completamente claro el cómo satisface las necesidades de negocio planteadas y el por qué este sistema es una solución tecnológica: se resalta como solución administrativa y de control (sobre todo en el módulo *Estadísticas*).

El desarrollo del software es una disciplina que se ha ido formando a partir de la experiencia en proyectos reales y sigue desarrollándose, puesto que cada vez surgen nuevas necesidades que requieren soluciones de software más complejas.

En esta carrera universitaria de la cual estoy saliendo como ingeniero enfocado al desarrollo de software, se busca la formación de profesionistas capaces de analizar problemas y diseñar soluciones a dichos problemas. Así que, en el software como en los demás campos, la ingeniería busca generar soluciones tecnológicas para luego implementarlas.

En resumen, puedo concluir que este trabajo no pude haberlo realizado sin los conocimientos previos que obtuve, de una u otra forma, de todas las asignaturas que llevé a lo largo de la carrera; así como los puntos de vista profesionales y personales, que cada profesor me transmitió. Todo aquello me ha ido formando un amplio panorama el cual me sirve ahora para afirmar que el desarrollo del sistema TRACKING es ingeniería pura.

Capítulo 6. Referencia bibliográfica

Libros

Ingeniería del Software Un enfoque práctico.
Roger Pressman.
McGraw Hill.
2002.

Ingeniería de Software.
Ian Sommerville.
Pearson Addison Wesley.
2005.

Diseño Orientado a Objetos con UML.
Raúl Alarcón.
Grupo Eidos.
2000.

Especificaciones de los requisitos del Software.
IEEE STD 830-1998.

A Pattern Language: Towns, Buildings, Construction.
Alexander, Christopher.
Oxford University Press.
1977.

Manual de Desarrollo Web con Grails.
Nacho Brito.
Ediciones Ágiles.
2009.

El Gran Libro de HTML5, CSS3 y Javascript.
Juan Diego Gauchat.
Marcombo.
2012.

Introducción a JavaScript.
Javier Eguíluz Pérez.
Creative Commons.
2009.

El maldito libro de los descarrilados.
Raúl Herrera.
Autoedición.
2011.

Scrum Manager.
Alexander Menzinsky.
Safe Creative.
2016.

Páginas Web

https://www.w3.org/Protocols/
(última visita: septiembre 2017)

http://www.agiledata.org/essays/mappingObjects.html
(última visita: septiembre 2017)

https://docs.python.org/3.5/howto/functional.html?highlight=paradigm
(última visita: septiembre 2017)

https://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concep
(última visita: septiembre 2017)

https://opensource.org/
(última visita: septiembre 2017)

https://www.thoughtco.com/sql-fundamentals-1019780
(última visita: septiembre 2017)

http://librosweb.es/libro/ajax/
(última visita: septiembre 2017)

https://www.w3.org/Style/CSS/
(última visita: septiembre 2017)

https://grails.org/
(última visita: septiembre 2017)

http://groovy-lang.org/
(última visita: septiembre 2017)

https://www.w3.org/html/
(última visita: septiembre 2017)

https://www.w3.org/MarkUp/
(última visita: septiembre 2017)

https://www.jetbrains.com/help/idea/grails.html
(última visita: septiembre 2017)

https://www.w3schools.com/js/
(última visita: septiembre 2017)

https://dev.mysql.com/doc/
(última visita: septiembre 2017)

https://www.apachefriends.org/es/index.html
(última visita: septiembre 2017)

http://www.re-orientation.com/manual-creacion-aplicacion-java-MVC
(última visita: septiembre 2017)