

CE3101 - Proyecto #1

Documentación técnica

Brayan Alfaro González
Julián Camacho Hernández
Kevin Zeledón Salazar
I Semestre 2021



Índice

Índice	1
Modelo conceptual	2
Modelo relacional	2
Estructuras de datos desarrolladas	3
Descripción detallada de la arquitectura desarrollada	5
Problemas sin solución	6
Problemas encontrados	6
Problema 1:	6
Problema 2:	6
Problema 3:	7
Conclusiones y recomendaciones	8
Conclusiones:	8
Recomendaciones:	8
Bibliografía	10



Modelo conceptual

- Ver en la penúltima página del documento.

Modelo relacional

- Ver en la última página del documento.

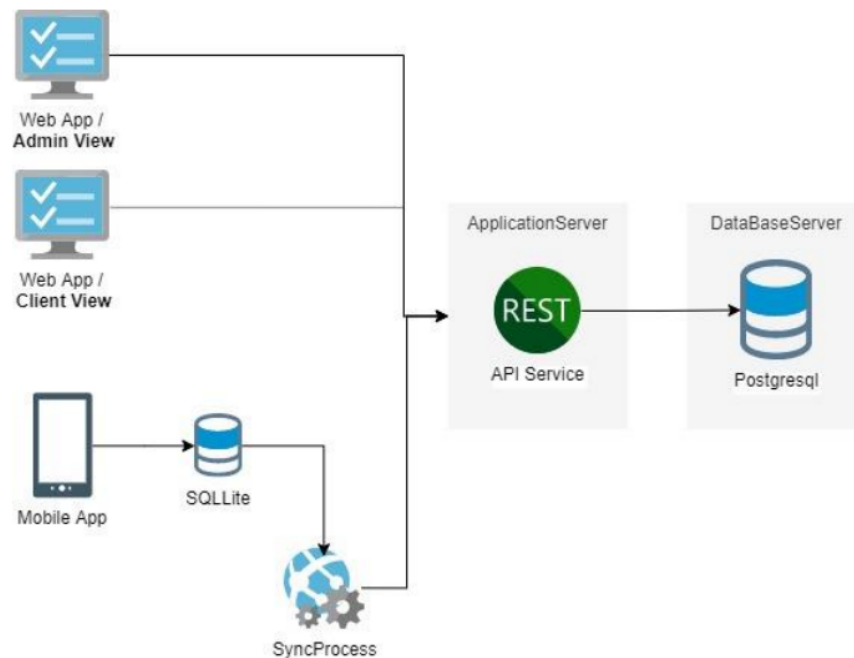
Estructuras de datos desarrolladas

En esta sección se presentan y se explica el uso de las estructuras de datos implementadas en el desarrollo de la tarea, específicamente las tablas de la base de datos.

Nombre	Descripción
ADMINISTRADOR	Contiene los datos de inicio de sesión de los usuarios administradores
APOSENTO	Contiene el nombre de un aposento de un usuario
CERTIFICADO_GARANTIA	Contiene los datos de la garantía de un dispositivo, como: fecha de compra de un dispositivo, tipo de dispositivo, numero de serie, correo del usuario, fecha de finalización de garantía y la duración de esta.
DISPOSITIVO	Contiene los datos de un dispositivo, como: el numero de serie, marca, descripción, precio, consumo eléctrico, tipo de dispositivo, cedula del distribuido, aposento en el que se encuentra, correo del usuario y el estado (encendido o apagado).
DISTRIBUIDOR	Contiene los datos de cédula jurídica y nombre del distribuidor. Los distribuidores brindan los dispositivos.
DISTRIBUIDOR_UBICACION	Contiene la región de un distribuidor.
DUENO_PASADO	Contiene el correo de un dueño histórico de un dispositivo, del cual contiene el número de serie.
FACTURA	La factura tiene un consecutivo, una fecha de compra y un precio.
FACTURA_DISPOSITIVO	Asocia a una factura con un dispositivo.
PEDIDO	El pedido contiene más datos que la factura. Contiene: monto, fecha de compra, consecutivo del pedido para el usuario, el número de serie del dispositivo, el correo del usuario y el id de la dirección del usuario.
REGION	Es especificada por un país y un continente.
TIPO_DISPOSITIVO	Define los tipos de dispositivos. Contiene un nombre, descripción, tiempo de garantía e imagen.

USO	Registra los tiempos de uso del usuario. Tiene una duración, fecha, y el número de serie del dispositivo asociado.
USUARIO	Corresponde al usuario final. Se registra el nombre completo, la contraseña y el correo.
USUARIO_DIRECCION	Guarda una dirección de un usuario. Contiene la ciudad, código postal, indicaciones y el correo electrónico del usuario, así como un identificador.
USUARIO_UBICACION	Guarda la región en la que se ubica un usuario.

Descripción detallada de la arquitectura desarrollada



El software para este proyecto consta de dos aplicaciones web, que son la vista de administrador y la vista de cliente. Mediante estas apps, los usuarios clientes y administradores son capaces de gestionar y acceder a toda la funcionalidad del software. Estas se conectan a un API que se encuentra en un ApplicationServer, que se encarga de procesar la información e interactuar al ser conectado con una base de datos Postgresql.

Por otro lado, se encuentra una aplicación móvil para los usuarios, que se conecta a una base empujada SQLite. Mediante un proceso de sincronización, los datos en esta base se gestionan en el API y se almacenan finalmente en la base de datos principal.

Problemas sin solución

- El tamaño de la imagen de fondo en la appweb del cliente a veces no es lo suficientemente larga para las dimensiones del dispositivo. Se intentaron usar diferentes configuraciones de css pero nunca se encontró una solución al problema.

Problemas encontrados

- **Problema 1:**
 - Descripción: En el API, en los métodos HTTP del controlador de dispositivos, se presentó un inconveniente al intentar retornar estructuras de tablas que se generan por medio de sentencias SQL como el join. Esto debido a que no existía un tipo de retorno que se adecuara a la nueva estructura.
 - Intentos de solución: Se intentó retornar un json que no correspondiera con ningún modelo establecido, pero no fue una solución funcional.
 - Solución encontrada: La solución encontrada fue crear una nueva clase modelo que tuviera como atributos las columnas de interés de la tabla resultado de realizar la combinación. De esa manera es posible poner este tipo como valor de retorno y así corresponder los datos con los esperados en el frontend.
 - Recomendaciones y conclusiones:
 - Se recomienda el uso de clases que modelan estructuras, tanto en el backend como en el frontend, para que la interacción entre ambos sea más ordenada.
 - Bibliografía consultada:
 - H., & Stoev, I. How can I get the count of a list in an Entity Framework model without including/loading the entire collection?.. [En línea]. Disponible en: <https://stackoverflow.com/questions/50471436/how-can-i-get-the-count-of-a-list-in-an-entity-framework-model-without-including>. [Accesado: 3- May- 2021].
- **Problema 2:**
 - Descripción: En la barra de navegación de la appweb del cliente, inicialmente no cambiaban de color los ítems cuando se encontraba abierta dicha página.
 - Intentos de solución: Se intentó utilizar diferentes opciones y clases de bootstrap para hacerlo funcionar correctamente, pero ninguna medida con bootstrap surtió efecto.

- Solución encontrada: La solución aplicada consistió en aplicar una clase condicional, usando las directivas de angular, la cual cambia el color de los elementos del navbar cuando la dirección actual es la asociada al elemento.
- Recomendaciones y conclusiones:
 - Se recomienda adquirir un conocimiento profundo y completo de los frameworks y tecnologías de diseño web, como son html, css, bootstrap, angular, entre otros, para así poder implementar funcionalidades que se comporten correctamente, de manera que no se deban de probar múltiples soluciones insatisfactorias.
 - Se concluye angular es una herramienta muy poderosa, ya que permite programar diferentes comportamientos tanto a nivel de componentes, en los archivos de typescript, como a nivel de elementos de html.
- Bibliografía consultada:
 - "Angular NgClass", *Angular.io*, 2021. [En línea]. Disponible: <https://angular.io/api/common/NgClass>. [Accesado: 28- Apr- 2021].

● Problema 3:

- Descripción: Cuando se realizaba una solicitud de get usuario en el api, no se devuelvían las tablas relacionadas al usuario, que se incluyen en la clase de c# que lo modela, como usuario_ubicacion y usuario_direccion, y sus respectivos valores en el json eran nulos.
- Intentos de solución: No hubo intentos de solución sin éxito.
- Solución encontrada: Para solventar este problema, se tuvo que declarar explícitamente, en el método de get, que se debía de incluir dicha información asociada mediante el uso del método de "Include".
- Recomendaciones y conclusiones:
 - Se recomienda investigar apropiadamente el funcionamiento de entity framework para conocer su comportamiento esperado, y así disminuir la cantidad de errores de semántica.
- Bibliografía consultada:
 - "Loading Related Data - EF Core", *Docs.microsoft.com*, 2021. [En línea]. Disponible: <https://docs.microsoft.com/en-us/ef/core/querying/related-data/>. [Accesado: 11- May- 2021].


Conclusiones y recomendaciones

Conclusiones:

- Se concluye que una manera rápida y eficiente de conectar una base de datos con un API es por medio del uso de Entity Framework en .NET. Esto ya que Entity Framework facilita la conexión con la base de datos, así como el modelado de los objetos y el manejo de distintos errores. Además de que este provee una serie de métodos para realizar queries en la base de datos, siguiendo una lógica similar a las instrucciones de SQL.
- Se concluye que postgresql es un gestor de bases de datos fiable y eficaz, ya que fue posible conectarlo al API de manera sencilla y nunca generó ningún problema difícil de solucionar.
- Asimismo, se concluye que Angular es un framework de desarrollo de web con muy amplias capacidades debido a la versatilidad de su arquitectura y a la gran cantidad de paquetes que posee, como http, routing, materials, entre otros. Además, esto se demostró gracias a la facilidad y naturalidad con la que se pudo implementar la aplicación web del proyecto.
- Se concluye también que la librería .NET provee de una gran cantidad de paquetes, y en particular, que provee de las herramientas necesarias para la creación de API's escalables a nivel profesional. Esto dado que permite y fomenta buenas prácticas como la creación de controladores y modelos.
- Además, el framework de Ionic provee una rápida experiencia de desarrollo y es el way to go para aplicaciones con funcionalidad nativa pequeña y, como es el caso de esta tarea, que ya tienen una contraparte web en alguna librería de javascript, como lo es Angular, pues permite una mejor integración y a la vez que todos los miembros del equipo sean multifuncionales y puedan trabajar en cualquier parte del proyecto.

Recomendaciones:

- Para la generación de reportes, se recomienda el uso de librerías especializadas, como DevExpress, ya que hacen posible la rápida generación de reportes visualmente llamativos que de otra manera serían muy difíciles de generar

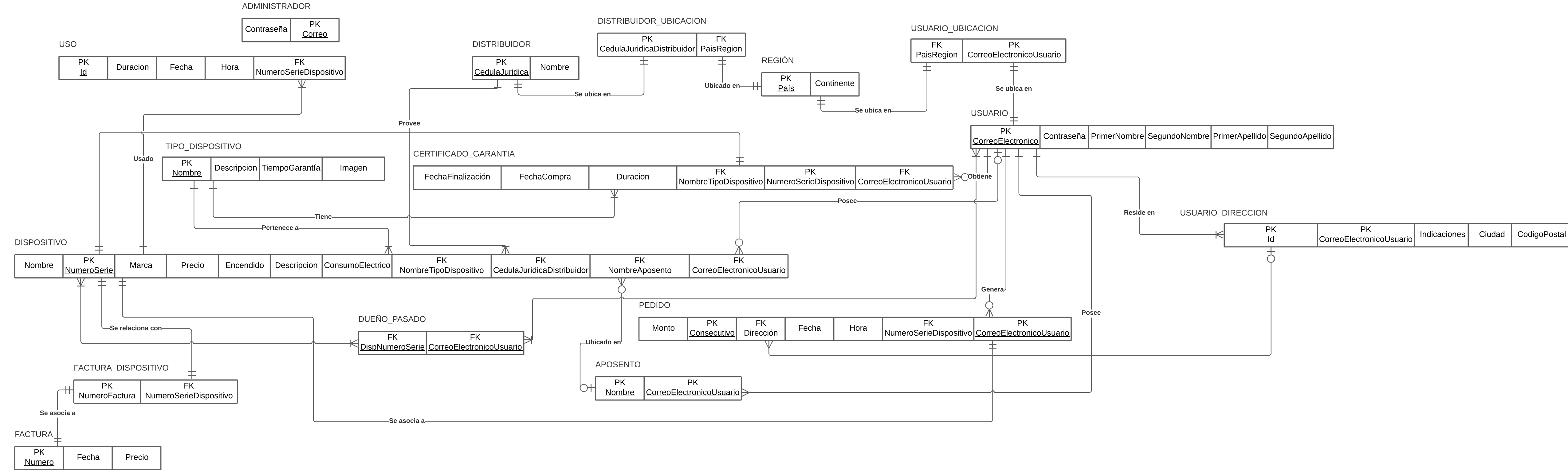
- 
- Se recomienda el uso de cookies al almacenar los datos de usuario en un proyecto de angular o node.js, esto ya que existen paquetes de npm muy fáciles de usar y que simplifican mucho el almacenamiento de datos del usuario.
 - Se recomienda realizar el diseño de las páginas web con antelación a la implementación para poder enfocarse meramente en asuntos de programación al momento de implementarla, y no tener que realizar tantos cambios en el código al no existir un diseño definido.



Bibliografía

- [1] "Angular", *Angular.io*. [En línea]. Disponible en: <https://angular.io/guide/component-interaction#parent-and-children-communicate-via-a-service>. [Accesado: 26- Mar- 2021].
- [2] "HTML 5.2", *W3.org*, 2021. [En línea]. Disponible en: <https://www.w3.org/TR/html52/>. [Accesado: 06- Mar- 2021].
- [3] ".NET documentation", *Docs.microsoft.com*, 2021. [En línea]. Disponible en: <https://docs.microsoft.com/en-us/dotnet/>. [Accesado: 20- Mar- 2021].

Modelo Relacional



Modelo Conceptual

