



# Certificado Profesional en Programación: Desarrollo Full-Stack con MERN

Autor: Iván Alejandro Fernández Gracia<sup>1</sup>

Profesores: John R. Williams<sup>2</sup> Abel Sanchez<sup>3</sup> Carolina Barreiro<sup>4</sup>

<sup>1</sup>Fullstack y Mobile Developer. Ingeniero Civil Mecanico, Universidad de Santiago de Chile.

<sup>2</sup>Profesor de Ingeniería de la Información en el Departamento de Ingeniería Civil y Ambiental del MIT.

<sup>3</sup>Científico investigador y Director Ejecutivo del Centro de Datos Geoespaciales del MIT.

<sup>4</sup>Digital Animation Engineer Project Management Specialist de la Universidad Panamericana, Mexico.

22 de abril de 2022

# Contenido

- 1 Introducción y Objetivos
- 2 Three-Tiered Applications
- 3 Arquitectura Frontend
- 4 Autenticación
- 5 Arquitectura Backend
- 6 Funcionalidades de la Aplicación
- 7 Implementacion del Proyecto
- 8 Continous Integration Delivery CI/CD
- 9 Extras: Cyber Security, UI, Roles
- 10 Referencias

# Contenido

- 1 Introducción y Objetivos
- 2 Three-Tiered Applications
- 3 Arquitectura Frontend
- 4 Autenticación
- 5 Arquitectura Backend
- 6 Funcionalidades de la Aplicación
- 7 Implementacion del Proyecto
- 8 Continous Integration Delivery CI/CD
- 9 Extras: Cyber Security, UI, Roles
- 10 Referencias

## Objetivos

- 1 **Evaluar** los conocimientos sobre la construcción, testeo y despliegue de una aplicación web **MERN** stack, **API** de backend con Express, **React**, pipeline de integracion continua (Continuous Integration) / entrega continua (Continuous Delivery) (**CI/CD**), interaccion con **base de datos**, **cybersecurity** y mucho mas.
- 2 Crear una **aplicación bancaria** donde los usuarios puedan registrarse para realizar **retiros, depositos, pagos**, etc.

# Contenido

- 1 Introducción y Objetivos
- 2 Three-Tiered Applications
- 3 Arquitectura Frontend
- 4 Autenticación
- 5 Arquitectura Backend
- 6 Funcionalidades de la Aplicación
- 7 Implementacion del Proyecto
- 8 Continous Integration Delivery CI/CD
- 9 Extras: Cyber Security, UI, Roles
- 10 Referencias

# Diagrama Three-Tiered con Tecnologias

## Tecnologias y motivo

- **Frontend:** El diseño UI/UX del sitio web es implementado con el framework **React**.
- **Backend:** El servidor que maneja las **peticiones de recursos** de los clientes y la **seguridad de los datos** se implementa con framework **Express**. Este servidor se conecta con 2 bases de datos.
- **Database:** La base de datos en la nube **MongoAtlas** se encarga de la persistencia de los datos y **Redis** del almacenamiento en cache de tokens para la seguridad de la app.



# Contenido

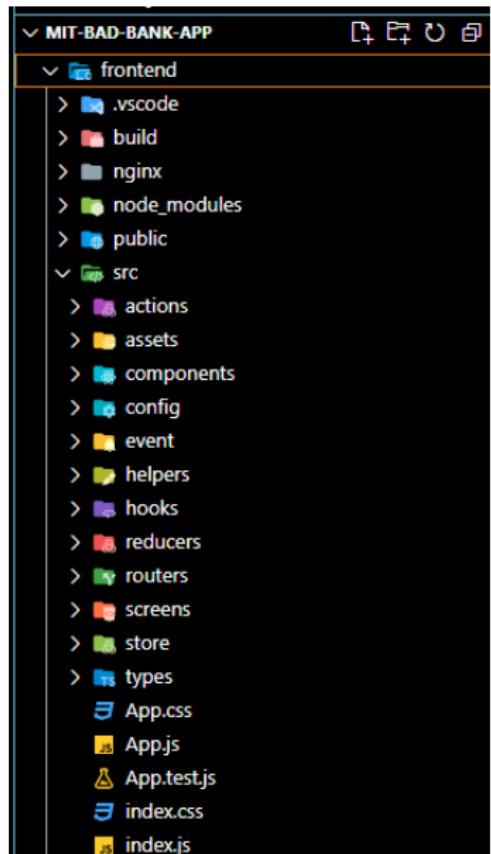
- 1 Introducción y Objetivos
- 2 Three-Tiered Applications
- 3 Arquitectura Frontend
- 4 Autenticación
- 5 Arquitectura Backend
- 6 Funcionalidades de la Aplicación
- 7 Implementacion del Proyecto
- 8 Continous Integration Delivery CI/CD
- 9 Extras: Cyber Security, UI, Roles
- 10 Referencias

# Framework y Estructura de Carpetas

## React y Carpetas

Se utiliza **React** como framework para el frontend de la app. La ilustración de la derecha muestra la estructura de carpetas. La **carpeta /scr** contiene subcarpetas que dividen la **lógica** del proyecto con un cierto orden:

- **Screens**: Vistas principales o contenedores de muchos componentes.
- **Component**: Componentes que cumplen con una sola funcionalidad.
- **Hooks**: Funciones que te permiten “enganchar” el estado de React y el ciclo de vida desde componentes de función.
- **Routers**: Componentes de navegación
- **Actions, Reducer, store**: Estado de la app REDUX
- **Helpers**: Encapsular funciones utilizadas en muchos componentes

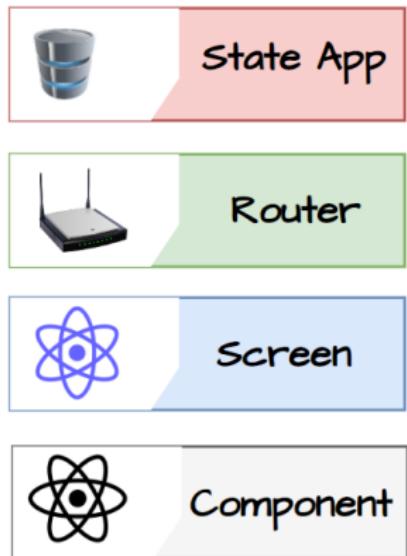


# Teoría: Diagrama Componentes React

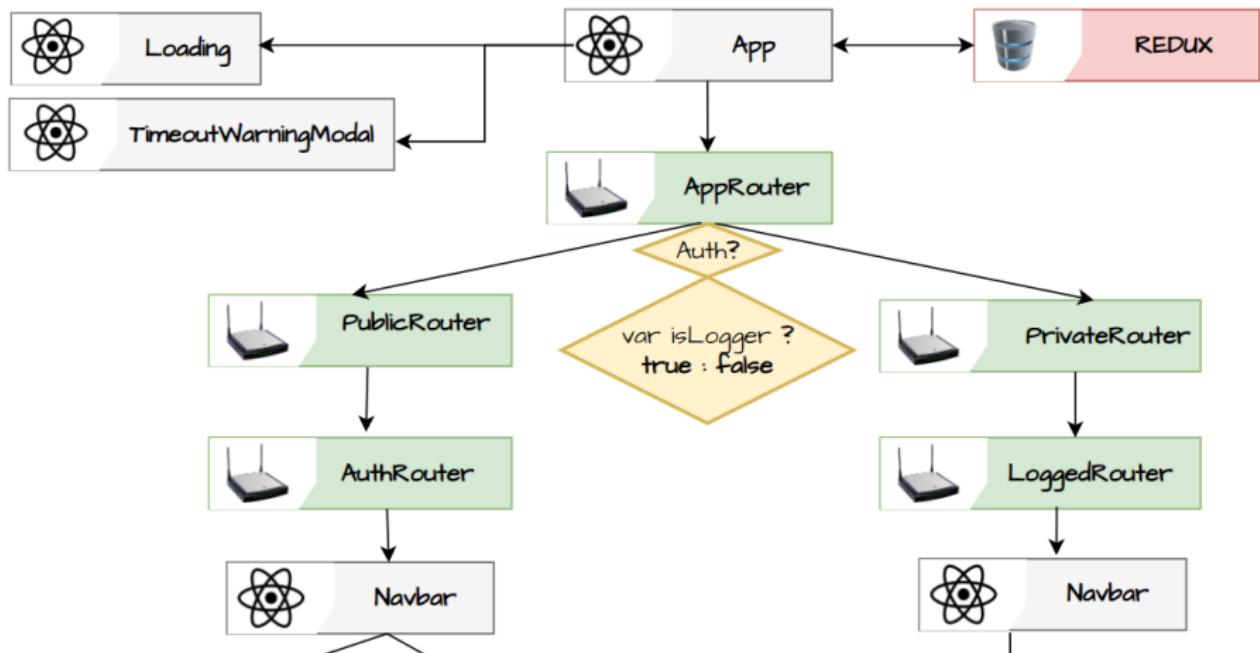
## Definición

Con el fin de explicar mejor la creacion UI y la estructura de la app, se les asigna definiciones a cada componente en React.

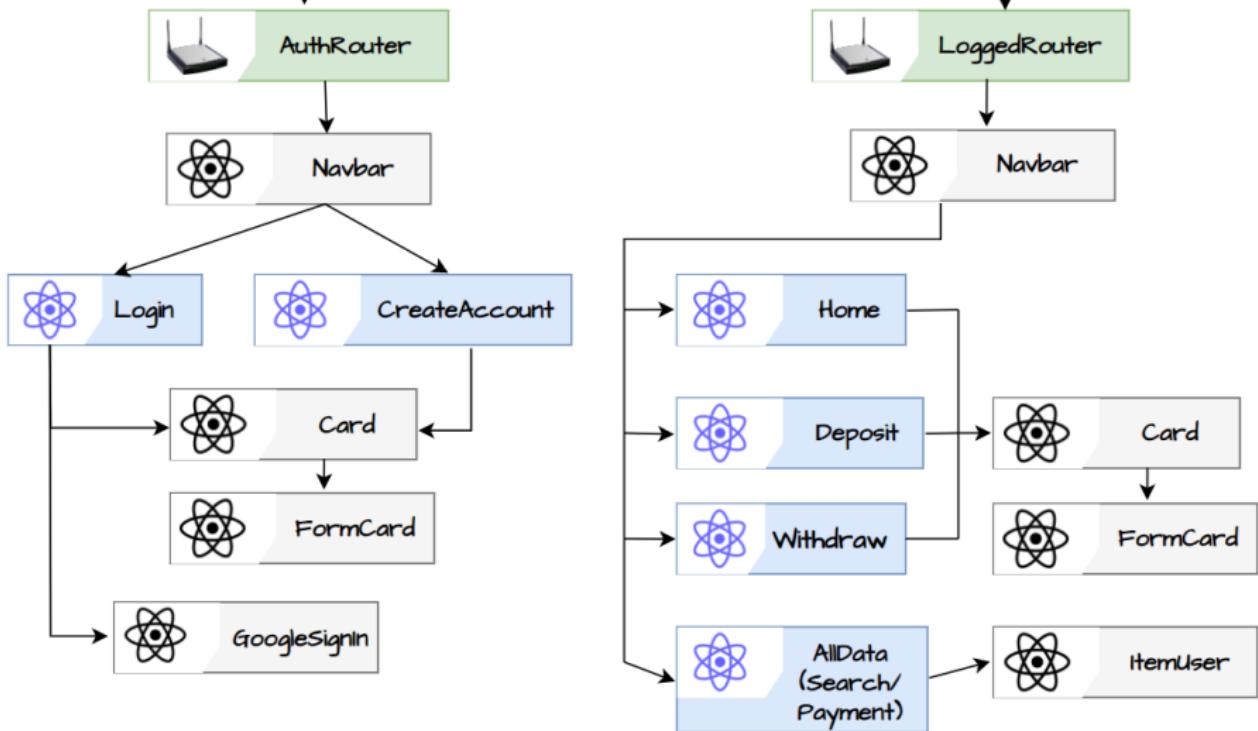
- **State App:** guardan el estado de uno o mas componente
- **Router:** permiten que rutas renderizan otros componente.
- **Screen:** componentes se utilizan como containers de otros componentes de bajo nivel.
- **Component:** componentes de bajo nivel y/o que desarrollan funcionalidades específicas.



# Diagrama Componentes React



# Diagrama Componentes React

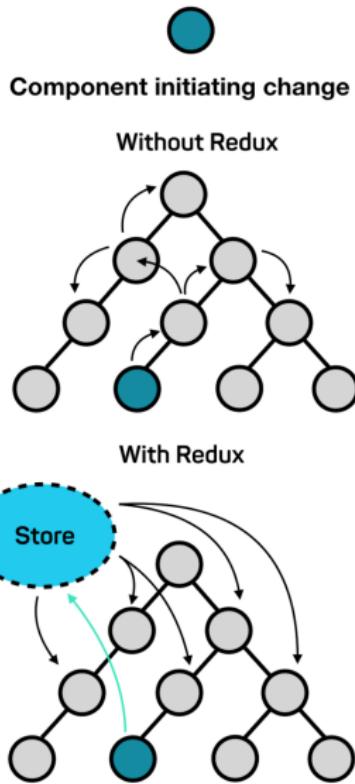


# Estado de la aplicacion: Redux

## Redux

Redux es un contenedor predecible del estado de aplicaciones JavaScript. Por medio de la librería react-redux se puede utilizar en React. Características:

- Estado **global** e inmutable
- Mayor **control** del estado de la aplicación y el flujo de datos
- Arquitectura **escalable** de datos

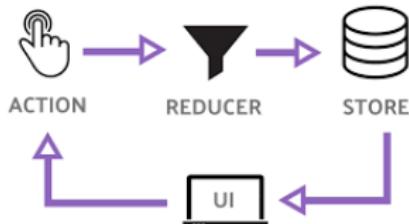


# Estado de la aplicacion: Redux

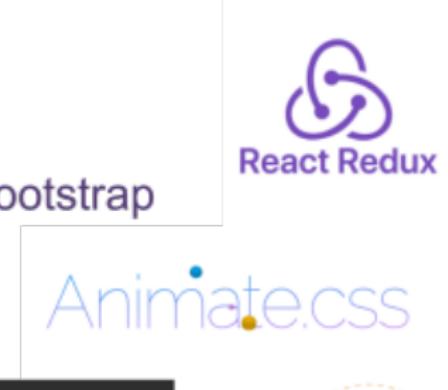
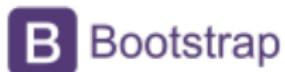
## Redux

Redux en patron Flux: **Store , State y Actions + Reducers**. Los estados son:

- **authUser**: Controla el estado del usuario que inicia sesión
- **ui**: Se encarga de mostrar y ocultar componentes en situaciones particulares, como loading al hacer peticiones al servidor.
- **allUsers**: Guarda una lista de usuarios con sus propiedades



State	Action	State	Diff	Trace	Test
Tree	Chart	Raw			
<b>authUser (pin)</b>					
checkingIsLogged (pin): <b>true</b>					
isLoggedIn (pin): <b>false</b>					
token_access (pin): <b>null</b>					
name (pin): <b>null</b>					
email (pin): <b>null</b>					
url_image (pin): <b>null</b>					
balance (pin): <b>null</b>					
<b>lastTransaction (pin)</b>					
amount (pin): <b>null</b>					
date (pin): <b>null</b>					
uid (pin): <b>null</b>					
type (pin): <b>null</b>					
► ui (pin): { isLoading: false }					
<b>allUsers (pin)</b>					
<b>users (pin)</b>					
▼ 0 (pin)					
name (pin): "Name"					
email (pin): "email@gmail.com"					
balance (pin): "10"					
url-image (pin): "https://bootdey.com/img/Content/avatar/avatar2.png"					



# Contenido

- 1 Introducción y Objetivos
- 2 Three-Tiered Applications
- 3 Arquitectura Frontend
- 4 Autenticación
- 5 Arquitectura Backend
- 6 Funcionalidades de la Aplicación
- 7 Implementacion del Proyecto
- 8 Continous Integration Delivery CI/CD
- 9 Extras: Cyber Security, UI, Roles
- 10 Referencias

# Métodos

## Autenticación

La autenticación es el proceso de identificar a los usuarios y garantizar que los mismos sean quienes dicen ser. Para lograr aquello en la app, se utilizan dos métodos:

- **Backend Server:** Comparación de contraseñas encriptadas en el servidor backend. Contraseñas guardadas en la base de datos.
- **Google Verify:** Verificación por medio de un servicio externo como lo es google sign in. por medio de un token de autorización que entrega google.

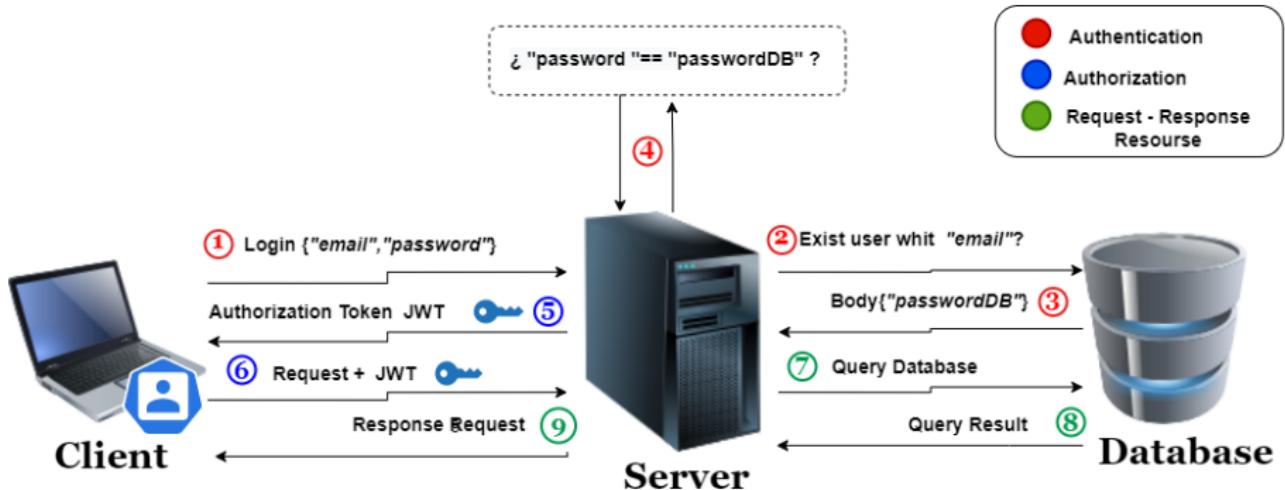


Backend Server

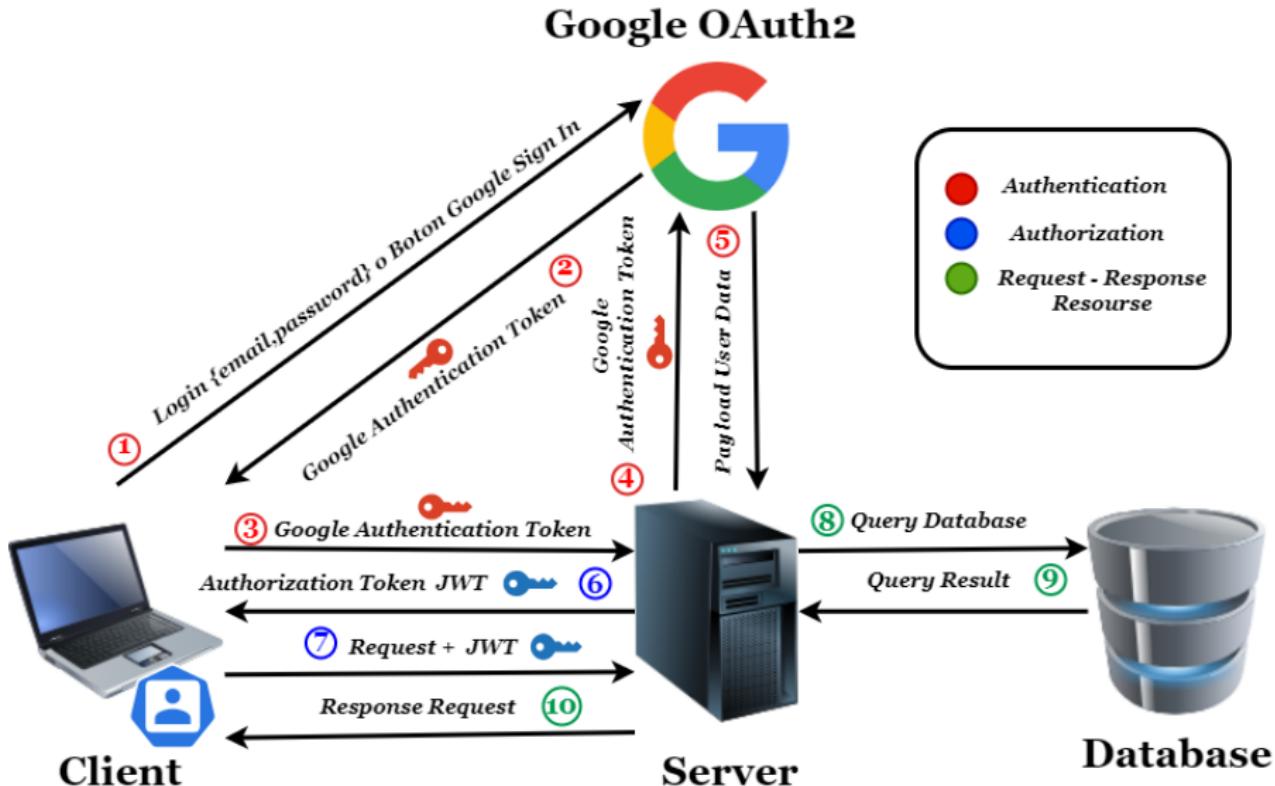


Google Sign In

# Diagrama: Backend Server



Comparación de autenticidad con contraseña encriptada unidireccional almacenada en la base de datos. La comparación se hace dentro del mismo servidor backend



# Contenido

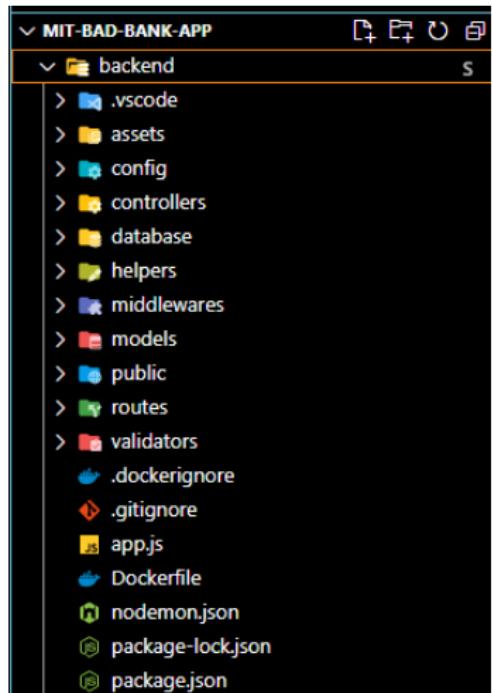
- 1 Introducción y Objetivos
- 2 Three-Tiered Applications
- 3 Arquitectura Frontend
- 4 Autenticación
- 5 Arquitectura Backend
- 6 Funcionalidades de la Aplicación
- 7 Implementacion del Proyecto
- 8 Continous Integration Delivery CI/CD
- 9 Extras: Cyber Security, UI, Roles
- 10 Referencias

# Framework y Estructura de Carpetas Server

## NodeJS y Carpetas

Se utiliza **Express** como framework de NodeJS para desarrollar el backend server de la app. La ilustracion de la derecha muestra la estructura de carpetas.

- **Models:** Existe una clase que configura el server y los modelos de la base datos MongoDB.
- **Database:** Configuracion de BD Mongo Atlas y Redis.
- **Routes:** Controla la respuesta a los endpoint segun su URI y verifica datos de entrada validos en cada peticion.
- **Controllers:** Controla la accion a realizar para cada endpoint.
- **Middlewares:** funcion que se puede ejecutar antes o despues del manejo de una ruta .



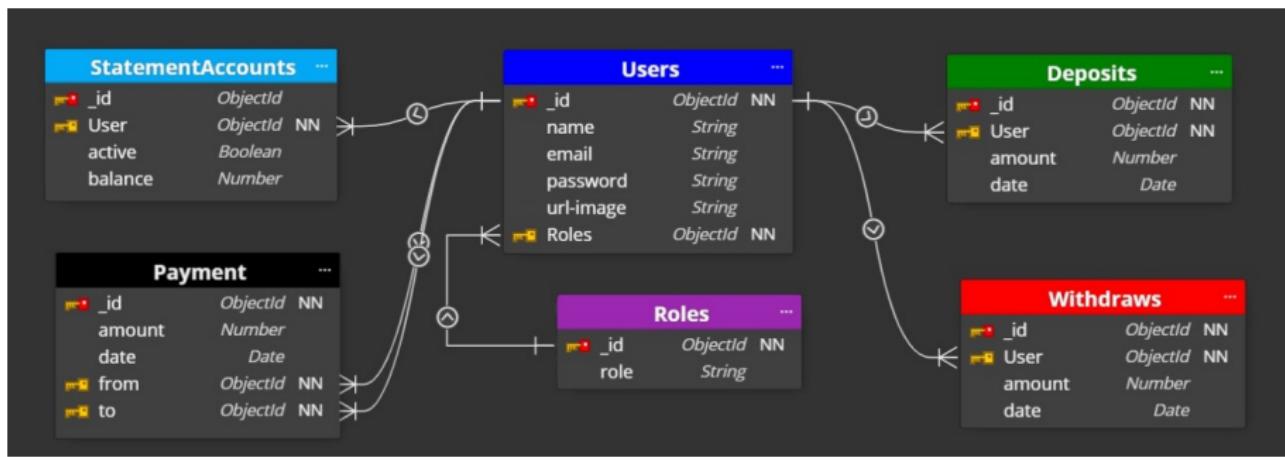
## Mongo Atlas

**MongoDB Atlas** es un servicio de **Cloud Database** (o Base de Datos en la Nube), que te permite crear y administrar tu BBDD Mongo desde cualquier lugar. Un esquema en **Mongoose** (ORM de MongoDB) es una estructura JSON que contiene información acerca de las propiedades de un documento. Se crean los siguientes esquemas:

- **Users:** Usuarios registrados para realizar acciones .
- **Roles:** Restringen que acciones pueden hacer los usuarios .
- **Deposits:** Deposito de dinero .
- **Withdraws:** Retiro de dinero .
- **StatementAccounts:** Estado de la cuenta de un usuario, como el balance de su dinero y si esta activa/bloqueada su cuenta .
- **Payments:** Pagos entre usuarios .



# Base de Datos: Mongo Atlas



Esquemas y conexiones creados en la base de datos Mongo Atlas

# Base de Datos: Mongo Atlas



Massachusetts  
Institute of  
Technology

## Users

```
_id: ObjectId("62563540816da2000ab53586")
name: "salsgiver"
email: "salsgiver.test@edbank.com"
role: ObjectId("623b4d9500abd95348e36062")
password: "$2a$10$ju94vL55M8bz3je1Fxq2B0el0u3me3ngvnpfrpxaf1u80TaaExL1Y6"
__v: 0
```

## Deposits

```
_id: ObjectId("6254fcdb113d6a7001974e409")
amount: 5932
date: 2022-04-12T04:15:15.656+00:00
user: ObjectId("6254fcdb113d6a7001974e407")
__v: 0
```

## Roles

```
_id: ObjectId("623b4d9500abd95348e36062")
role: "USER"
__v: 0
```

## StatementAccounts

```
_id: ObjectId("62562468eb66980019d9eecc")
active: true
balance: 3500
user: ObjectId("62563467cb66980019d9eecc")
__v: 0
```

## Withdraws

```
_id: ObjectId("62437f90911f322524c3fg9e")
amount: 6
date: 2022-03-19T23:52:16.733+00:00
user: ObjectId("62437c04dac81900e4780ba5")
__v: 0
```

## Payment

```
_id: ObjectId("625647e65256820019d48124")
amount: 1000
date: 2022-04-13T08:47:58.975+00:00
from: ObjectId("62562467eb66980019d9eecc")
to: ObjectId("6256354059abf1001a940e35")
__v: 0
```

Ejemplo de esquemas creados

## Redis Data Base

**Redis**, que significa Remote Dictionary Server, es un rápido almacén de datos clave-valor en memoria de código abierto. Por medio de **Redis-Commander** se puede visualizar la base de datos. Este proyecto se utiliza Redis para guardar los token temporales de autenticación y autorización. Las clave/valor que guarda en esta app son:

- **Clave:** Id único del usuario de MongoDB.
- **Valor:** JSONstringyf con un objeto que contiene Access Token y Refresh Token .



# API REST: End Point



Massachusetts  
Institute of  
Technology



Swagger

Supported by SMARTBEAR

## Auth Autentificacion y autorizacion

**POST** /api/auth/login Iniciar sesion con correo y contraseña.

**POST** /api/auth/createAccount Crear usuario.

**POST** /api/auth/google Iniciar sesion con token google oauth2.

**GET** /api/auth/logout Cerrar Sesion.

**POST** /api/auth/renewTokens Renovar tokens .

## csrf Proteccion contra ataques CSRF

**GET** /api/csrf/getCSRFToken Crear y enviar token CSRF.

**GET** /api/csrf/deleteCSRFToken borrar token CSRF.

## Deposit Depositar Dinerio al banco

**POST** /api/deposit/ Depositar dinero.

## Payment Pagos entre usuarios

**POST** /api/payment/ Depositar dinero.

## User Datos de los usuarios

**GET** /api/users/:term obtener usuarios con sus datos.

# Contenido

- 1 Introducción y Objetivos
- 2 Three-Tiered Applications
- 3 Arquitectura Frontend
- 4 Autenticación
- 5 Arquitectura Backend
- 6 Funcionalidades de la Aplicación
- 7 Implementacion del Proyecto
- 8 Continous Integration Delivery CI/CD
- 9 Extras: Cyber Security, UI, Roles
- 10 Referencias

# Funcionalidades

## Funcionalidades

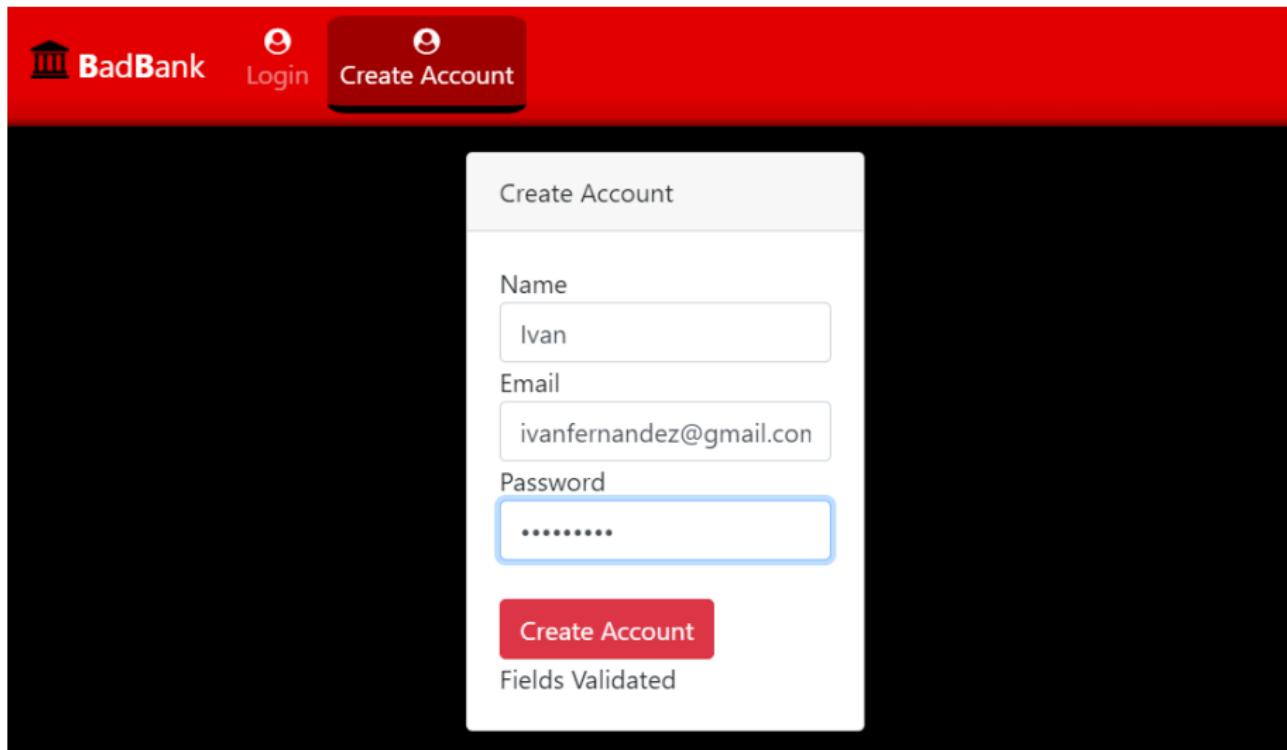
Las principales funcionalidades de la aplicación son las siguientes acciones que puede realizar el usuario o cliente:

- **Crear cuenta:** con una dirección de correo electrónico y una contraseña.
- **Iniciar Sesión:** el usuario puede iniciar sesión con una dirección de correo electrónico, contraseña o autenticación OAuth 2.0 Google Sign In .
- **Depositar:** El usuario puede depositar dinero.
- **Retirar:** El usuario puede retirar dinero.
- **Base de datos:** Información del usuario y de los otros registrados en base de datos.



# BadBank

# Crear cuenta: Email y Password



The image shows a screenshot of a web application interface titled "Create Account". The application has a red header bar. On the left side of the header is a logo for "BadBank" featuring a stylized building icon and the text "BadBank". To the right of the logo are two buttons: "Login" and "Create Account", where "Create Account" is highlighted with a red background and white text. The main content area is a white form with a thin gray border. At the top of the form is the title "Create Account". Below it are three input fields: "Name" containing the value "Ivan", "Email" containing "ivanfernandez@gmail.com", and "Password" containing a series of six dots (...). Below these fields is a red button labeled "Create Account". At the bottom of the form, the text "Fields Validated" is displayed.

Create Account

Name

Ivan

Email

ivanfernandez@gmail.com

Password

.....

Create Account

Fields Validated

# Login o Crear cuenta: Google Sign In

The image displays two side-by-side web pages. The left page is the 'Login Account' screen for 'BadBank'. It features a red header with a bank icon and the text 'BadBank'. Below the header are three buttons: 'Login' (highlighted in red), 'Create Account', and a third unlabelled button. The main form has 'Email' and 'Password' fields, both of which have error messages: 'Format Incorrect' and 'Less than 8 characters' respectively. A large red 'Login Account' button is at the bottom. A 'Acceder con Google' button is also present. The right page is a 'Sign In - Google Accounts' page. It shows a 'G' logo and the text 'Iniciar sesión con Google'. Below this, it says 'Selecciona una cuenta para ir a mitbadbank.com'. It lists an account for 'Ivan Fernandez' with the email 'ivan.fernandez.g@usach.cl'. There is also a link 'Usar otra cuenta'.

BadBank

Login Create Account

Login Account

Email

Enter Email

Format Incorrect

Password

Enter Password

Less than 8 characters

Login Account

Acceder con Google

Sign In - Google Accounts - Go... accounts.google.com/gsi/select?client\_id=...

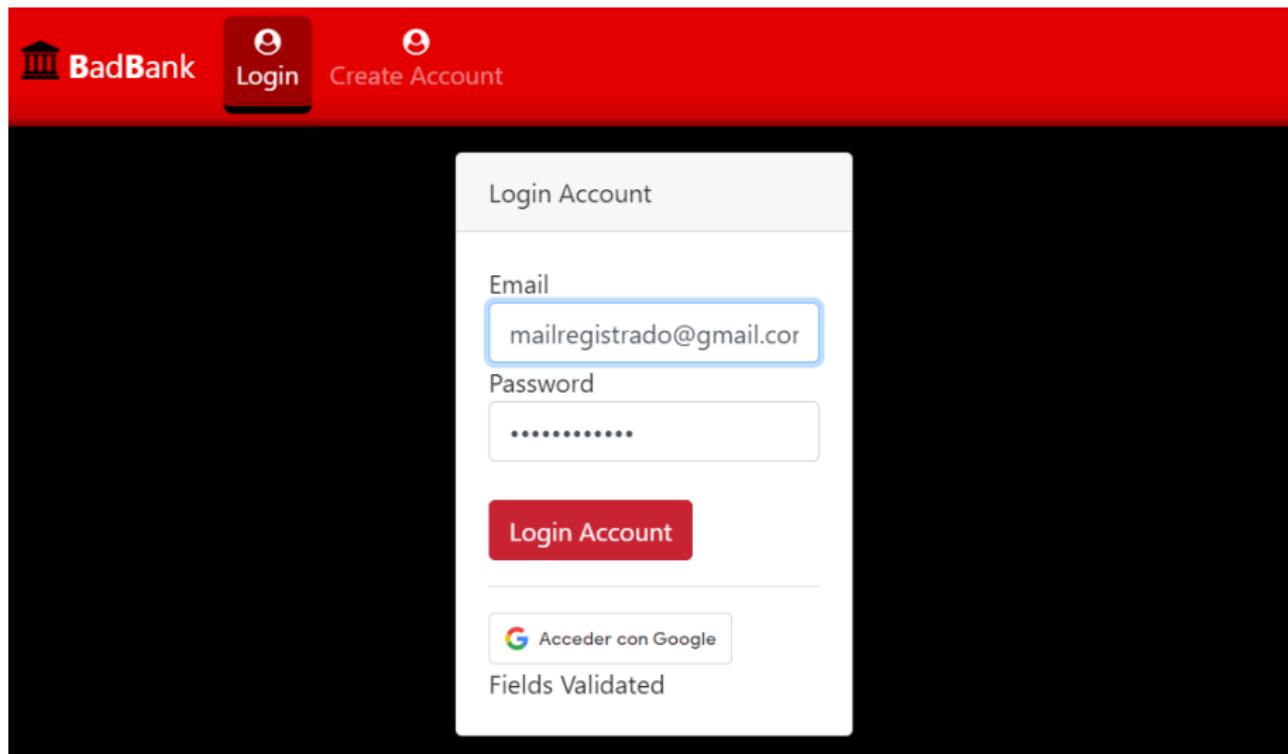
Iniciar sesión con Google

Selecciona una cuenta para ir a mitbadbank.com

Ivan Fernandez  
ivan.fernandez.g@usach.cl

Usar otra cuenta

# Login cuenta: Email y Password



The screenshot shows the BadBank login interface. At the top left is the logo 'BadBank' with a bank building icon. To its right are 'Login' and 'Create Account' buttons. The main area is a white card titled 'Login Account'. It contains two input fields: 'Email' with the value 'mailregistrado@gmail.com' and 'Password' with masked input. Below these is a large red 'Login Account' button. Further down is a 'Acceder con Google' button featuring the Google logo. A message 'Fields Validated' is displayed at the bottom of the card.

BadBank

Login Create Account

Login Account

Email

mailregistrado@gmail.com

Password

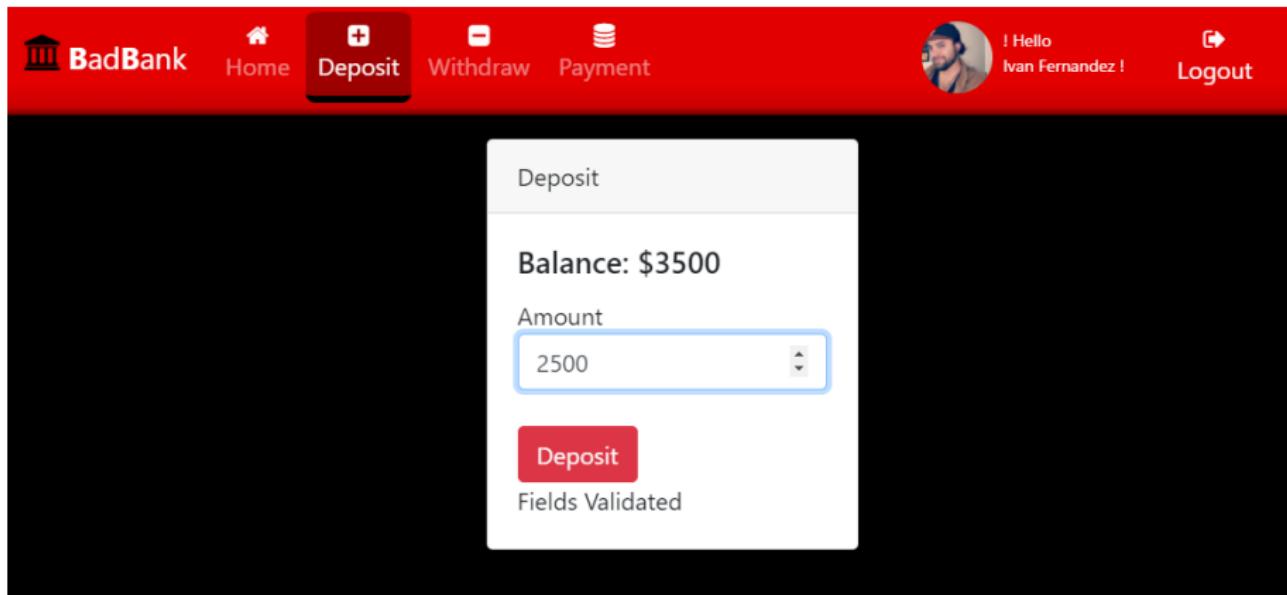
\*\*\*\*\*

Login Account

Acceder con Google

Fields Validated

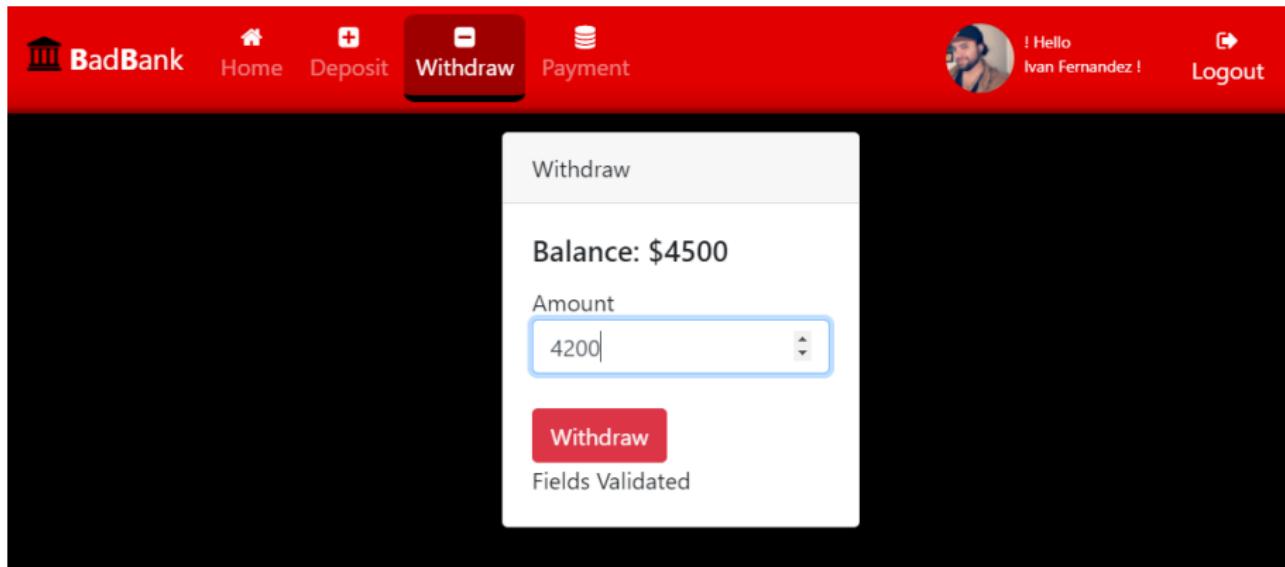
# Depositar



The screenshot shows a mobile application interface for a bank. At the top, there is a red navigation bar with the following items from left to right: a bank icon labeled "BadBank", a "Home" button with a house icon, a "Deposit" button with a plus sign and a banknote icon (which is highlighted in black), a "Withdraw" button with a minus sign and a banknote icon, a "Payment" button with a stack of coins icon, a user profile picture of a man with a beard, the greeting "Hello Ivan Fernandez!", and a "Logout" button with a logout icon.

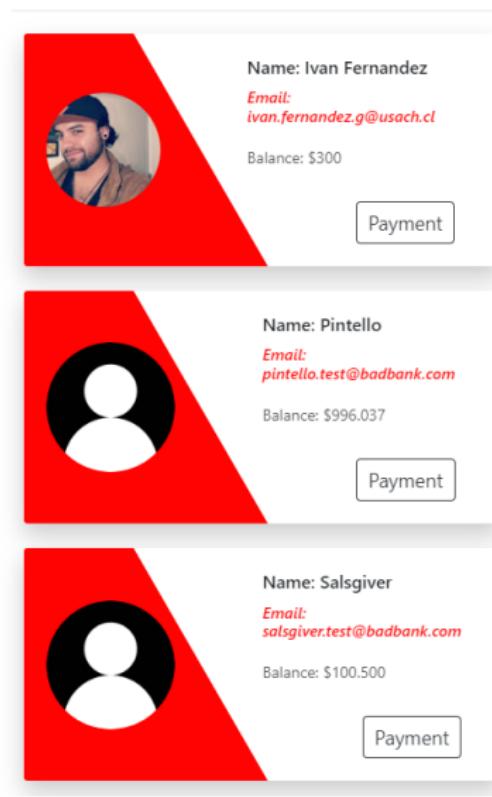
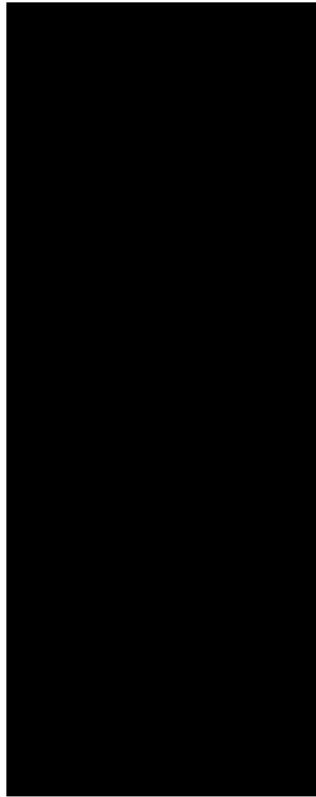
The main content area has a black background. In the center, there is a white rectangular form titled "Deposit". Inside the form, the text "Balance: \$3500" is displayed. Below it, the label "Amount" is followed by an input field containing the value "2500". At the bottom of the form is a red button labeled "Deposit". Below the button, the text "Fields Validated" is shown in gray.

## Retirar



The image shows a screenshot of a web application interface for a bank named "BadBank". The top navigation bar is red and features the bank's logo (a stylized building icon), the name "BadBank", and several menu items: "Home", "Deposit", "Withdraw", and "Payment". On the right side of the top bar, there is a user profile picture of a man with the text "Hello Ivan Fernandez!" and a "Logout" button. The main content area has a black background and contains a white modal window titled "Withdraw". Inside the modal, the text "Balance: \$4500" is displayed above an input field labeled "Amount" containing the value "4200". Below the input field is a large red button labeled "Withdraw". At the bottom of the modal, the text "Fields Validated" is shown.

# Persistencia base de datos Mongo Atlas



The image shows three user profiles from a mobile application interface:

- Ivan Fernandez**:  
Email: [ivan.fernandez.g@usach.cl](mailto:ivan.fernandez.g@usach.cl)  
Balance: \$300  
[Payment](#)
- Pintello**:  
Email: [pintello.test@badbank.com](mailto:pintello.test@badbank.com)  
Balance: \$996.037  
[Payment](#)
- Salsgiver**:  
Email: [salsgiver.test@badbank.com](mailto:salsgiver.test@badbank.com)  
Balance: \$100.500  
[Payment](#)



# Persistencia base de datos Mongo Atlas

```
_id: ObjectId("62563467cb66980019d9eecc")
name: "Ivan Fernandez"
email: "ivan.fernandez.g@usach.cl"
password: "googleAuth"
url_image: "https://lh3.googleusercontent.com/a-/AOh14G
role: ObjectId("623b4d0500abd05348e36062")
__v: 0
```

```
_id: ObjectId("62563540816da2001ab535b5")
name: "Salsgiver"
email: "salsgiver.test@badbank.com"
role: ObjectId("623b4d0500abd05348e36062")
password: "$2a$10$ju94VL5SW8z3je1Fxq28UeKu3WoZnQvpfnf
__v: 0
```

```
_id: ObjectId("6256354059abf1001a940e35")
name: "Pintello"
email: "pintello.test@badbank.com"
role: ObjectId("623b4d0500abd05348e36062")
password: "$2a$10$E9LWPTi5feWQ/XehXX0MaumAy6C2LA3XM7oNT
__v: 0
```

```
_id: ObjectId("62563468eb66980019d9eecd")
active: true
balance: 300
user: ObjectId("62563467cb66980019d9eecc")
__v: 0
```

```
_id: ObjectId("62563540816da2001ab535b6")
active: true
balance: 100500
user: ObjectId("62563540816da2001ab535b5")
__v: 0
```

```
_id: ObjectId("6256354059abf1001a940e36")
active: true
balance: 996037
user: ObjectId("6256354059abf1001a940e35")
__v: 0
```

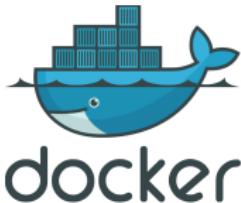
# Contenido

- 1 Introducción y Objetivos
- 2 Three-Tiered Applications
- 3 Arquitectura Frontend
- 4 Autenticación
- 5 Arquitectura Backend
- 6 Funcionalidades de la Aplicación
- 7 Implementacion del Proyecto
- 8 Continous Integration Delivery CI/CD
- 9 Extras: Cyber Security, UI, Roles
- 10 Referencias

# Virtualizacion y Hosting

## Virtualizacion

Docker utiliza **contenedores** que incluyen todo lo necesario para que un software se ejecute. El contenedor **frontend** con Nginx con el sitio web estatico y un balanceador de carga para las peticiones, 2 contenedores **server** con NodeJS que maneja la logica de la app y se conecta con la nube MongoDB Atlas, el contenedor **redis** para guardar tokens de seguridad y el contenedor **redis-commander** para visualizar la base de datos Redis.



## Hosting

AWS Elastic Beanstalk permite implementar, escalar servicios y aplicaciones web. EB administra de manera automática la implementación: la capacidad, el equilibrio de carga y el escalado automático hasta la monitorización del estado de la aplicación. El instancia EC2 permite implementar contenedores docker.



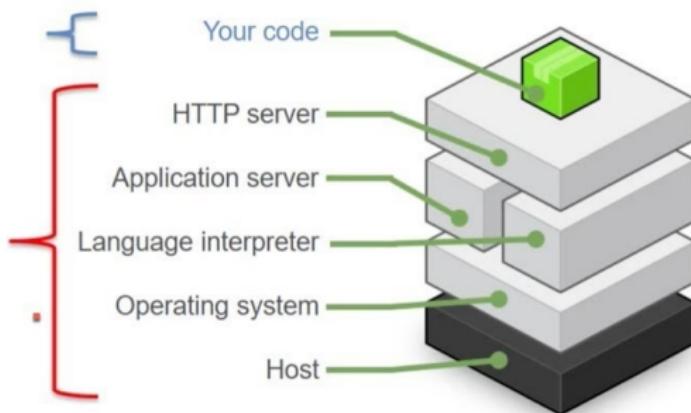
ELASTIC BEANSTALK

## Elastic Beanstalk

### On-instance configuration

Focus on building your application

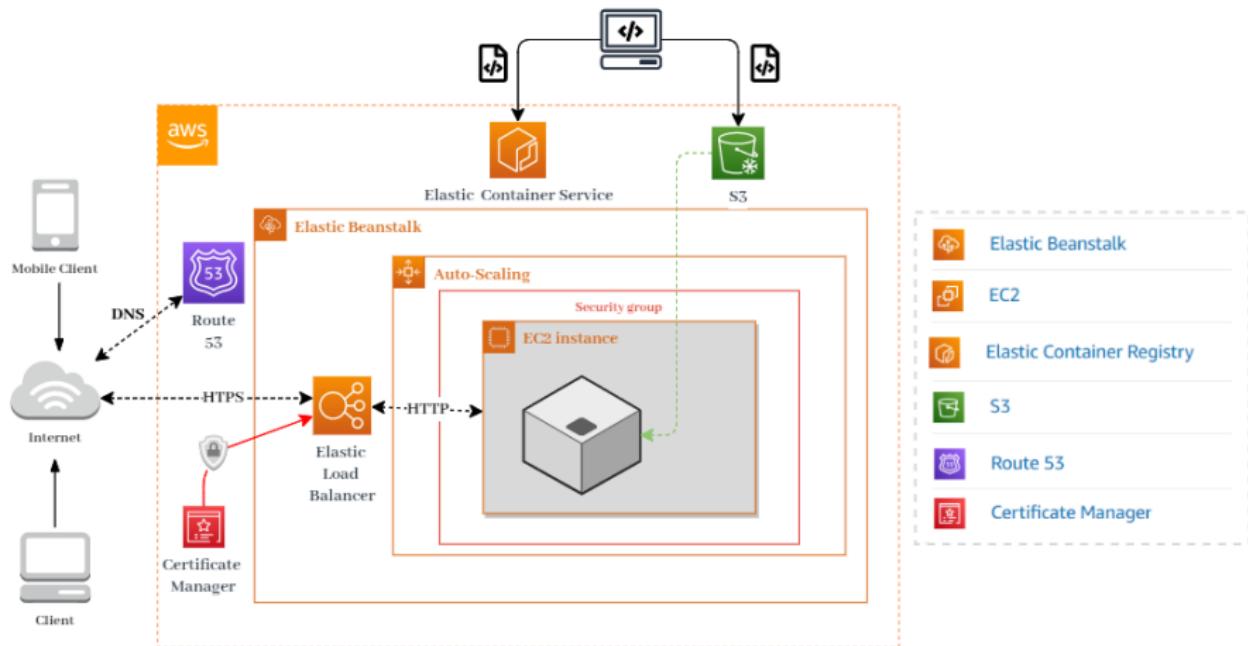
Elastic Beanstalk configures each Amazon EC2 instance in your environment with the components necessary to run applications for the selected platform. No more worrying about logging into instances to install and configure your application stack.



 Provided by you

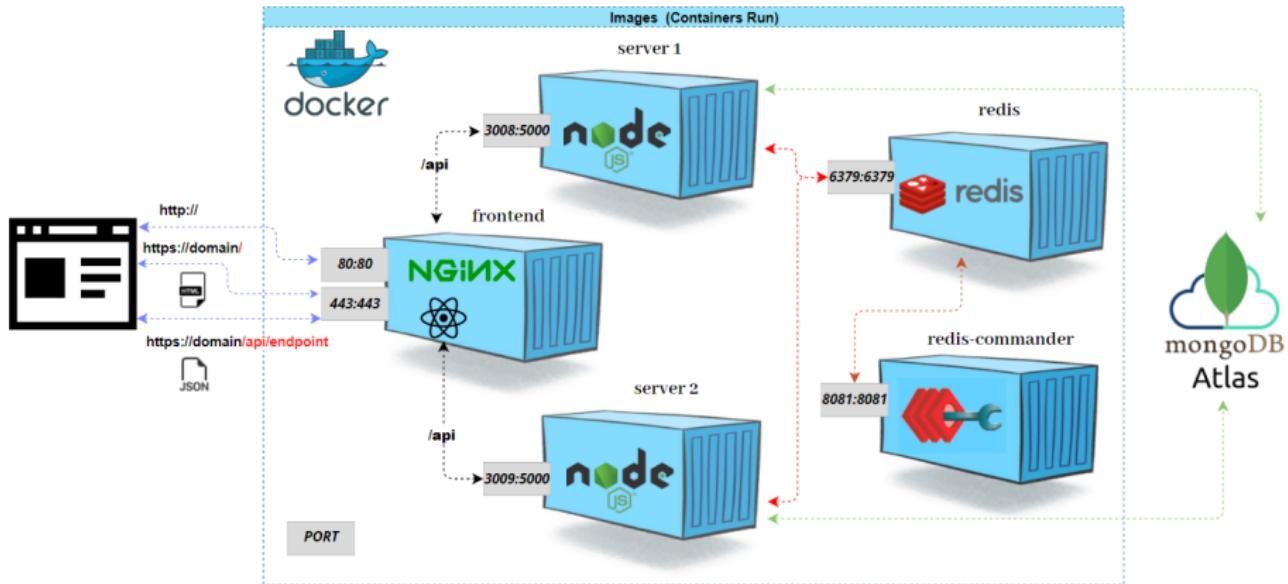
 Provided and managed by Elastic Beanstalk

# AWS Elastic Beanstalk



Servicios de AWS utilizados: **ECR** para registrar imágenes docker, **Route3** para DNS y dominio, **CM** para peticiones con SSL, **ELB** para peticiones http y https, **EC2** para la ejecución de contenedores docker y **S3** para almacenar los archivos(código) de la app en formato .zip

# Contenedores Docker y Flujo de Peticiones

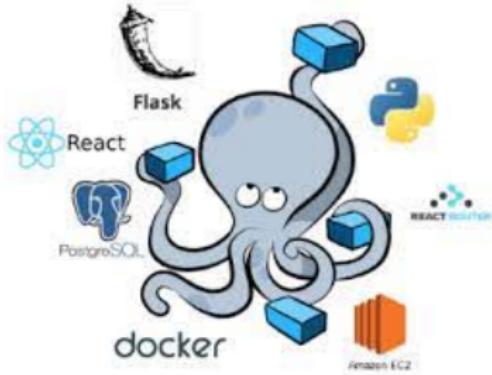


Contenedores Docker con sus **puertos** de escucha, **protocolos** de peticiones,  
**rutas** + base de datos Mongodb Atlas Cloud

# Docker Compose

## Multi-Contenedor

Compose es una herramienta para definir y ejecutar aplicaciones Docker de **contenedores múltiples**. Con Compose, utiliza un archivo .YML para configurar los servicios de su aplicación. Luego, con un solo comando, crea e inicia todos los servicios desde su configuración del archivo.

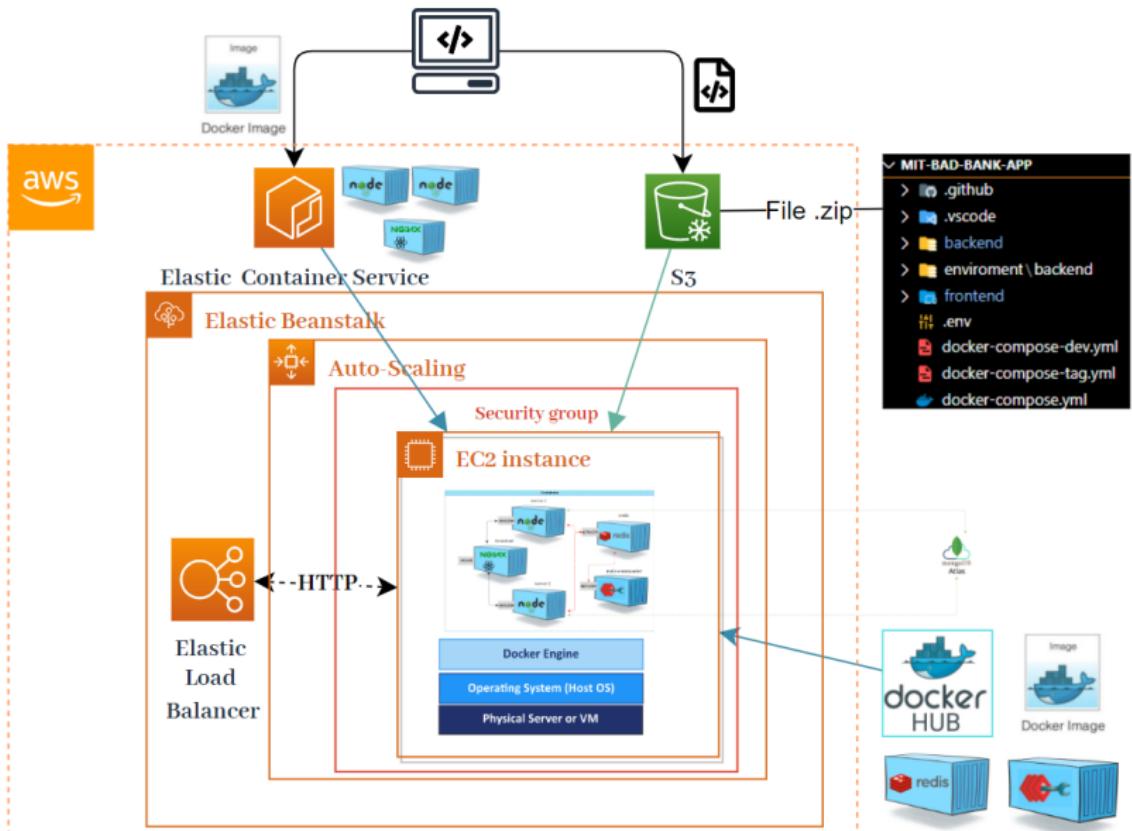


## Ejemplo

En la ilustración inferior se aprecia un extracto de el archivo .yml, donde se configura el contenedor **frontend** para su ejecución. Se detalla el **nombre de la imagen** y **ruta** del archivo **Dockerfile**, un archivo de **variables de entorno**, configuración de **reinicio**, **puertos** de escucha y de los **contenedores que depende**.

```
# version of docker-compose
version: '3.7'
# 'services' are equivalent to 'containers'
services:
  frontend:
    image: ${DOCKER_REGISTRY}/frontend:latest
    build:
      context: ./frontend
      env_file: "./frontend/.env"
      restart: always
      environment:
        - NODE_ENV=${DOCKER_ENV_FRONTEND}
    ports:
      - "80:80"
      - "443:443"
    depends_on:
      - server1
      - server2
```

# Implementacion Docker + AWS EB



# Contenido

- 1 Introducción y Objetivos
- 2 Three-Tiered Applications
- 3 Arquitectura Frontend
- 4 Autenticación
- 5 Arquitectura Backend
- 6 Funcionalidades de la Aplicación
- 7 Implementacion del Proyecto
- 8 Continous Integration Delivery CI/CD
- 9 Extras: Cyber Security, UI, Roles
- 10 Referencias

## Continuous Integration (CI)

## Continuous Deployment (CD)

### Source Code Control

Lanzamientos de CI/CD pipeline basados en checkeos de código



**GitHub**

### Build & Test

Empieza automáticamente la construcción de archivos y el testeo



### Release Automation

Actualizar el repositorio de artefactos con los últimos artefactos/contenedores que hayan tenido éxito para mantener el registro y la accesibilidad.



### Deploy & Production

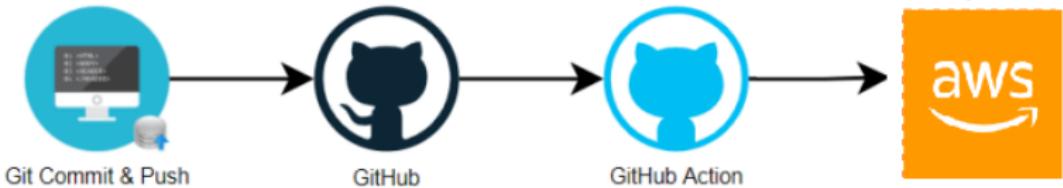
Desplegar aplicaciones en la fase de preparación y migrarlas a la fase de producción



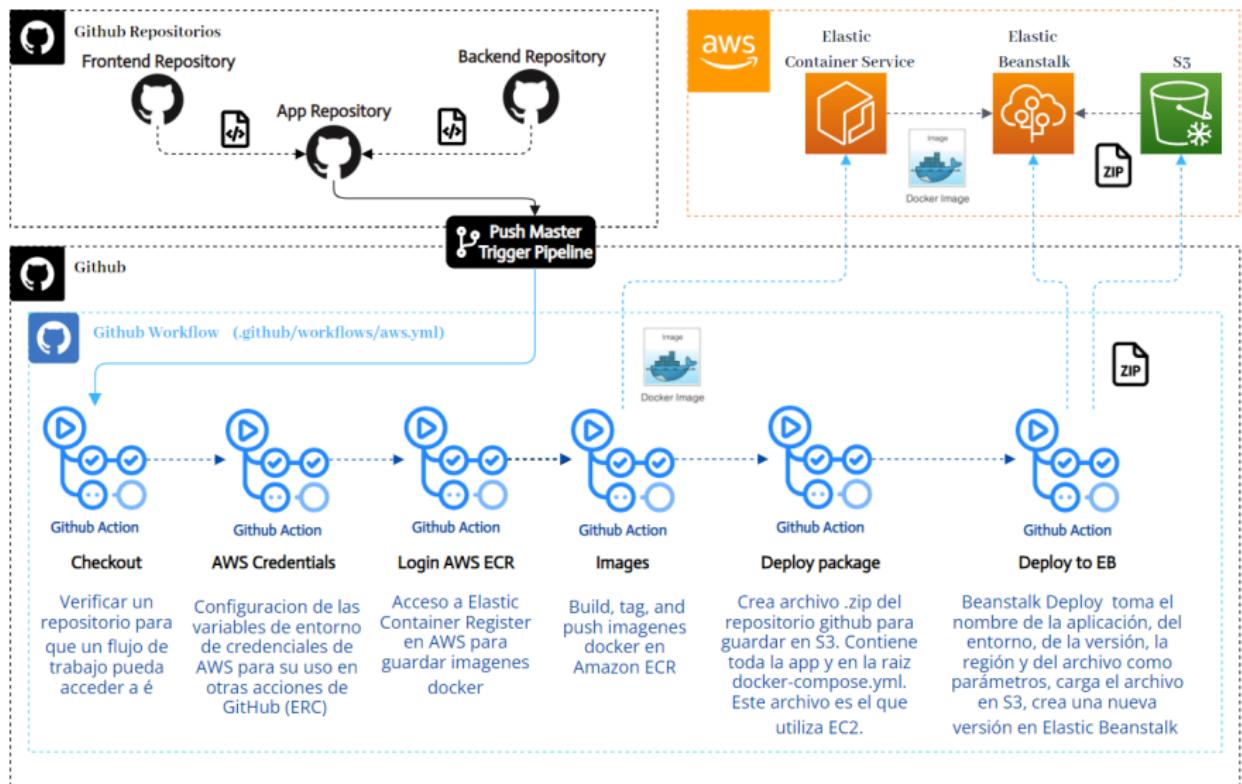
# Resumen pipeline

## Workflow de la aplicacion

- El desarrollador aloja en **repositorios en github** los códigos de la app con el sistema de **versiones git**. En cada nueva versión del código alojado, github **activa una acción**, la cual tiene el fin de **subir a producción la app a AWS**.



# Workflow Pipeline



# Contenido

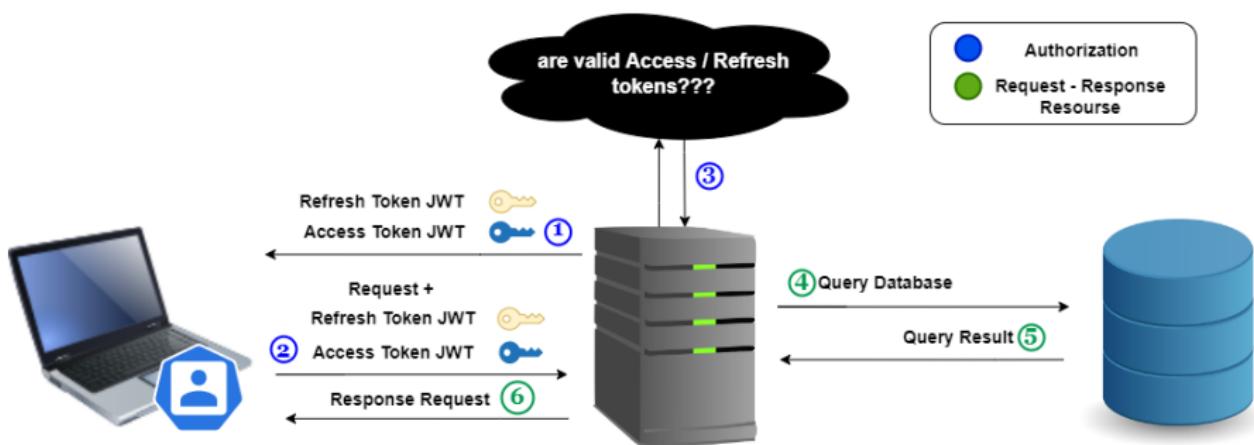
- 1 Introducción y Objetivos
- 2 Three-Tiered Applications
- 3 Arquitectura Frontend
- 4 Autenticación
- 5 Arquitectura Backend
- 6 Funcionalidades de la Aplicación
- 7 Implementacion del Proyecto
- 8 Continous Integration Delivery CI/CD
- 9 Extras: Cyber Security, UI, Roles
- 10 Referencias

# Cyber Security: AccessToken y Refresh Token

## Autorizacion y Seguridad

Con el fin de **proteger los datos** para que no sean enviados a personas no autorizadas, se crean 2 tokens. Estos se entregan después de la **autenticación** (iniciar sesión). Un **middleware** en el server verifica en cada petición que necesite autorización si estos tokens son válidos

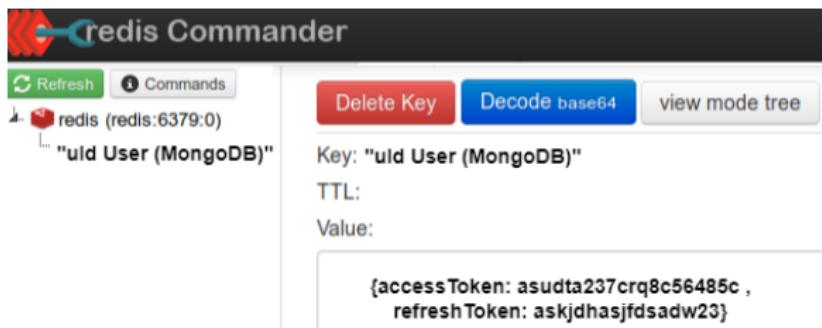
- **AccessToken:** Utilizado como **permiso para pedir recursos a la API**. Contiene id del usuario y expira rápidamente. Se guarda en **memoria**, es decir, en un estado o **variable JS**.
- **Refresh Token:** Se utiliza para **validar la creación de un Access Token** cuando este último expira. Contiene id del usuario y tiene un tiempo de expiración mayor al Access token. Para evitar ataques de hackers **cross-site scripting(XSS)** se guarda en una **cookie**.



# Cyber Security: AccessToken y Refresh Token

Name	Value	Size	HttpOnly	Secure	SameSite
token_refresh	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX...	185	✓	✓	Strict
_csrf-my-app	4_zfU5uGLo0Rjx6ezObf45ml	36	✓	✓	Strict

Refresh Token guardado en navegador como una cookie  
(Frontend)



Redis Commander

Commands

redis (redis:6379:0)

"uid User (MongoDB)"

Delete Key Decode base64 view mode tree

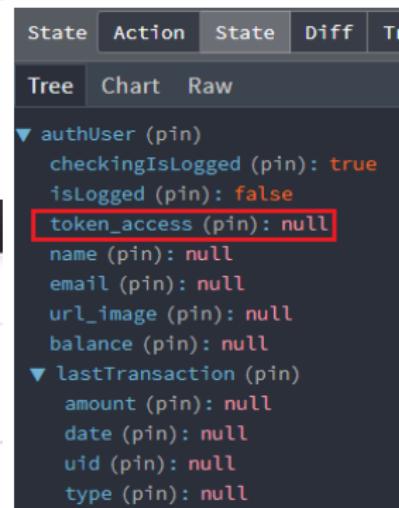
Key: "uid User (MongoDB)"

TTL:

Value:

```
{accessToken: asudta237crq8c56485c , refreshToken: askjdhasjfdsadw23}
```

Access y Refresh Token guardado en Redis para verificaciones de autorización en el server (backend)



State	Action	State	Diff	Tree
Tree	Chart	Raw		

authUser (pin)

checkingIsLoggedIn (pin): true

isLoggedIn (pin): false

token\_access (pin): null

name (pin): null

email (pin): null

url\_image (pin): null

balance (pin): null

lastTransaction (pin)

amount (pin): null

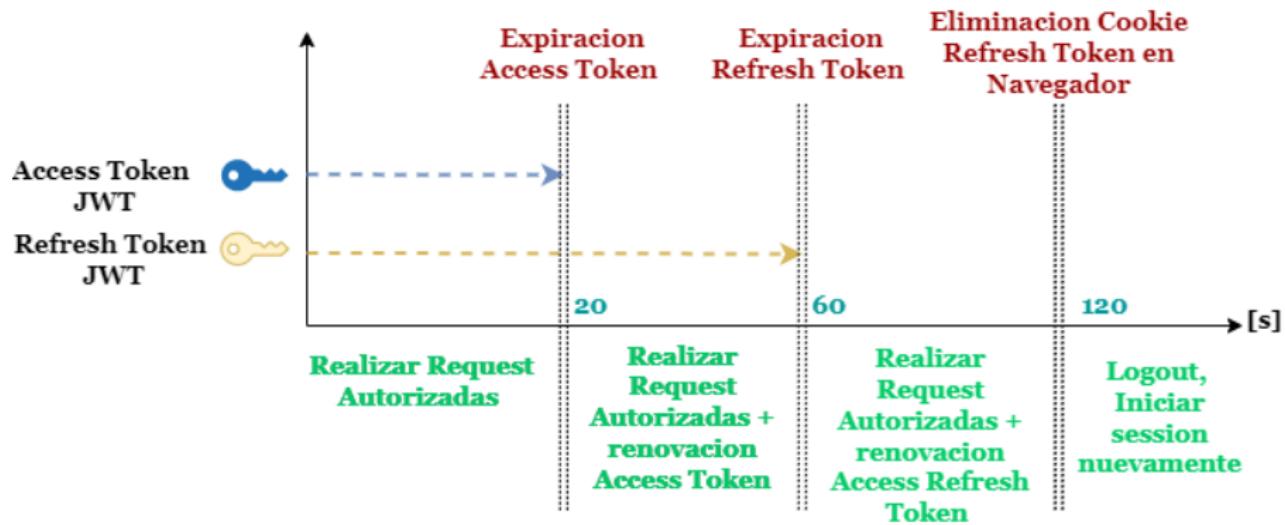
date (pin): null

uid (pin): null

type (pin): null

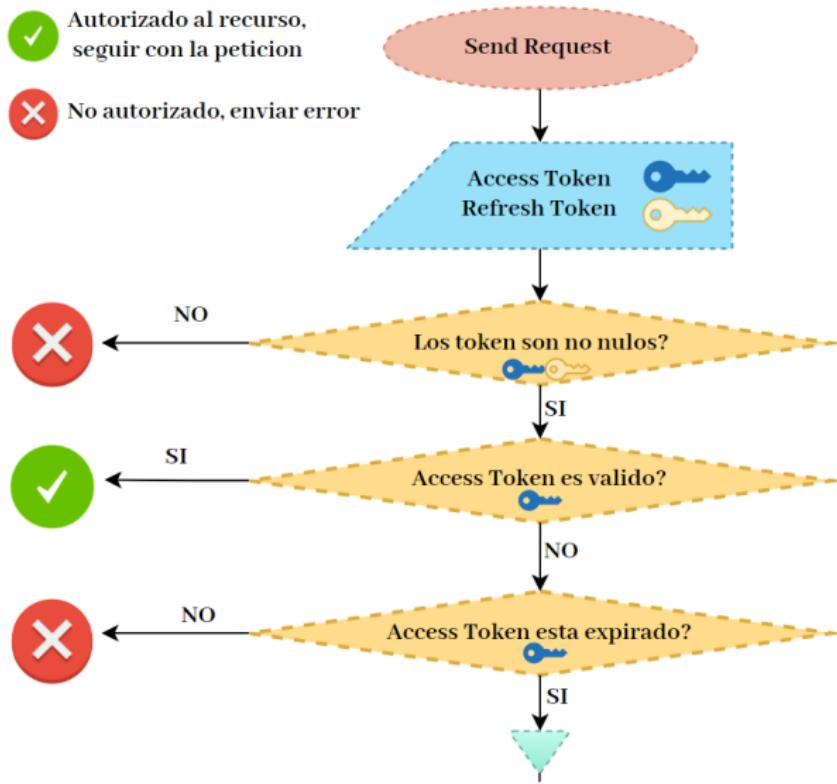
Access Token guardado en estado Redux en React  
(Frontend)

# Cyber Security: AccessToken y Refresh Token

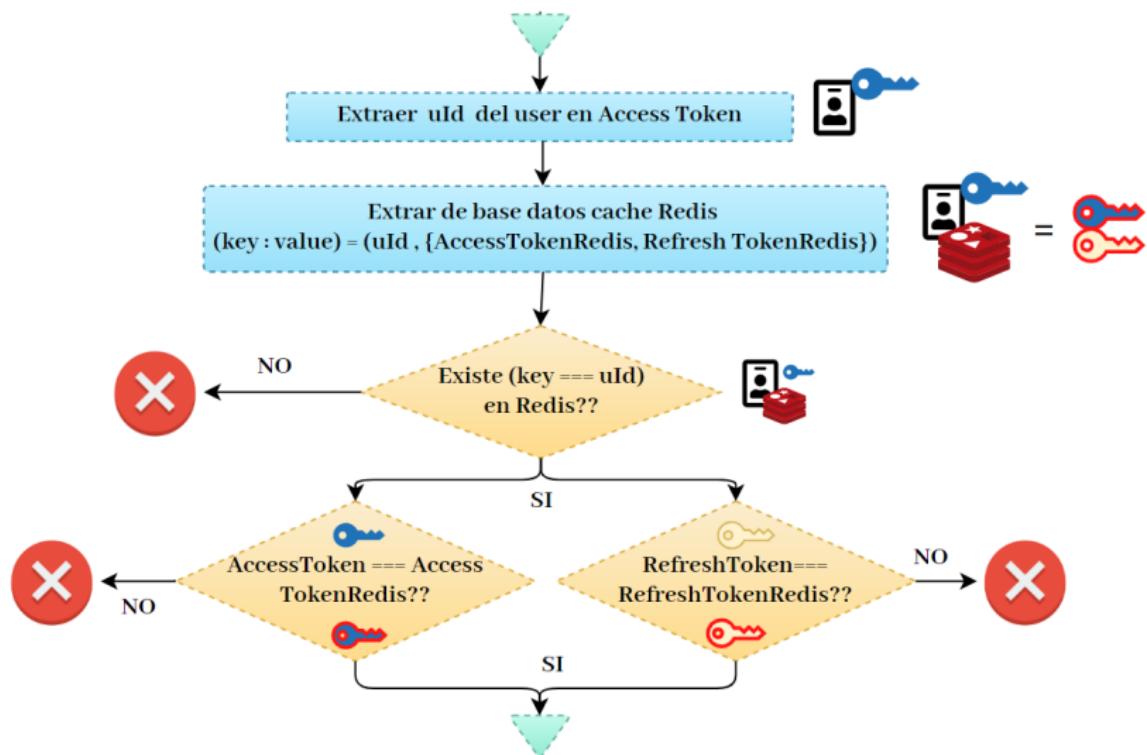


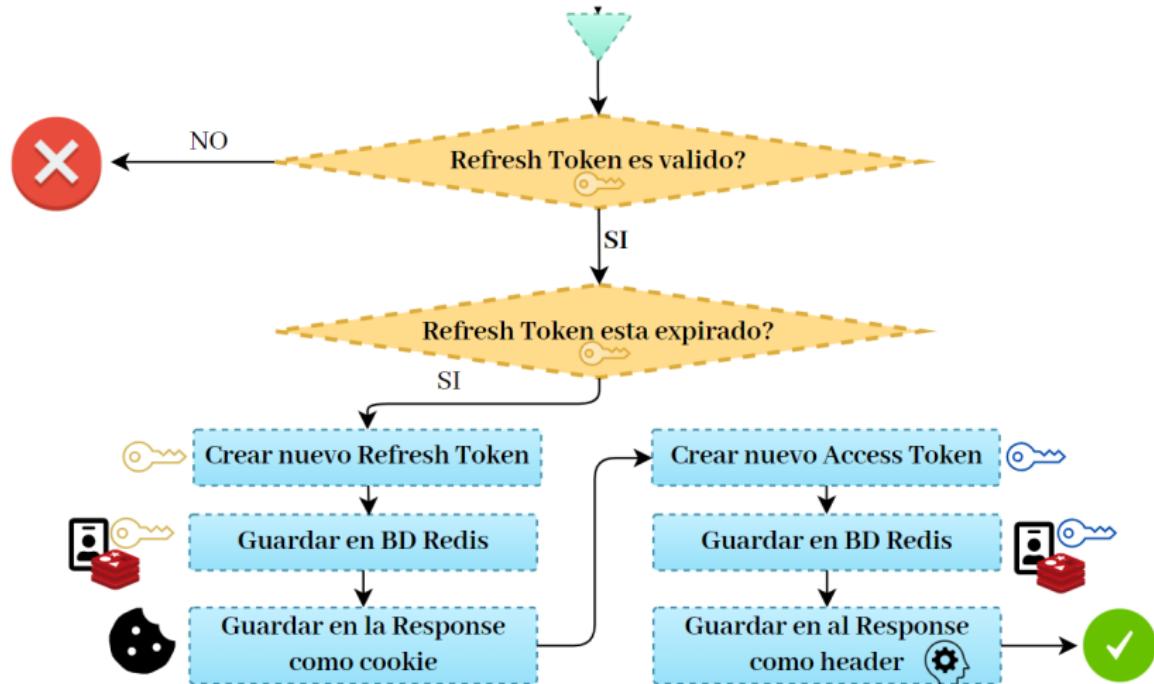
Estado para realizar peticiones a la API, renovación de tokens, tiempo de expiración de tokens y logout.

# Cyber Security: Middleware verificacion Tokens

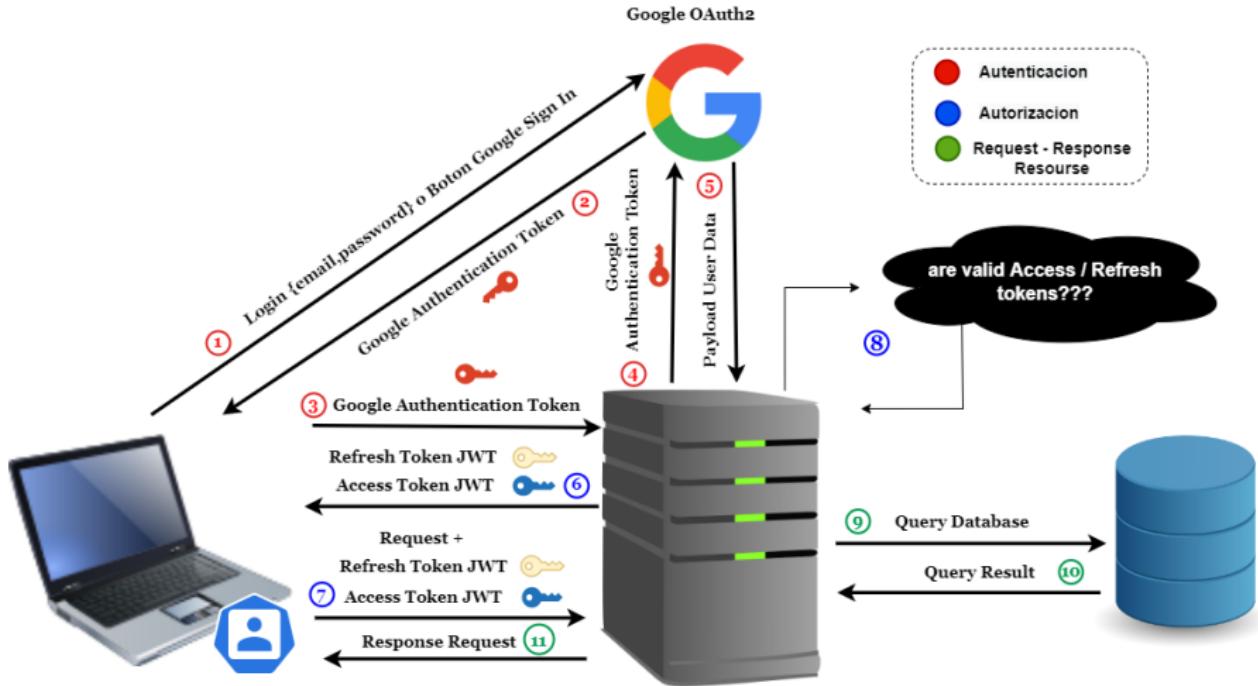


# Cyber Security: Middleware verificacion Tokens





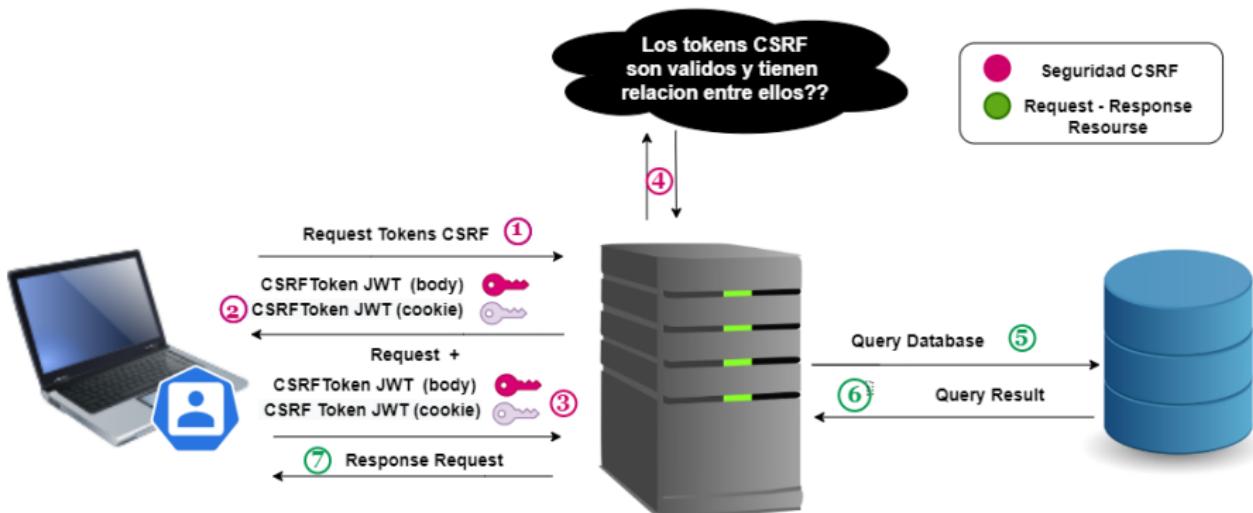
# Cyber Security: Autenticación y Autorización



# Cyber Security: CSRF Token

## Autorizacion y Seguridad

Con el fin de proteger los datos de un ataque Cross Site Request Forgery (CSRF) se crean 2 token. Uno es enviado en el **body** de una respuesta a un endpoint específico para pedir estos tokens y se guarda en un **tag en html** (recomendable dentro de <head>). El otro se envia por medio de una **cookie**. El **tiempo de expiración** de este token es 2 veces el Refresh Token (120 [s]). Un **middleware** en el server verifica que estos dos token tengan una relación válida cada vez que se envía una petición distinta a GET.



# Cyber Security: CSRF Token

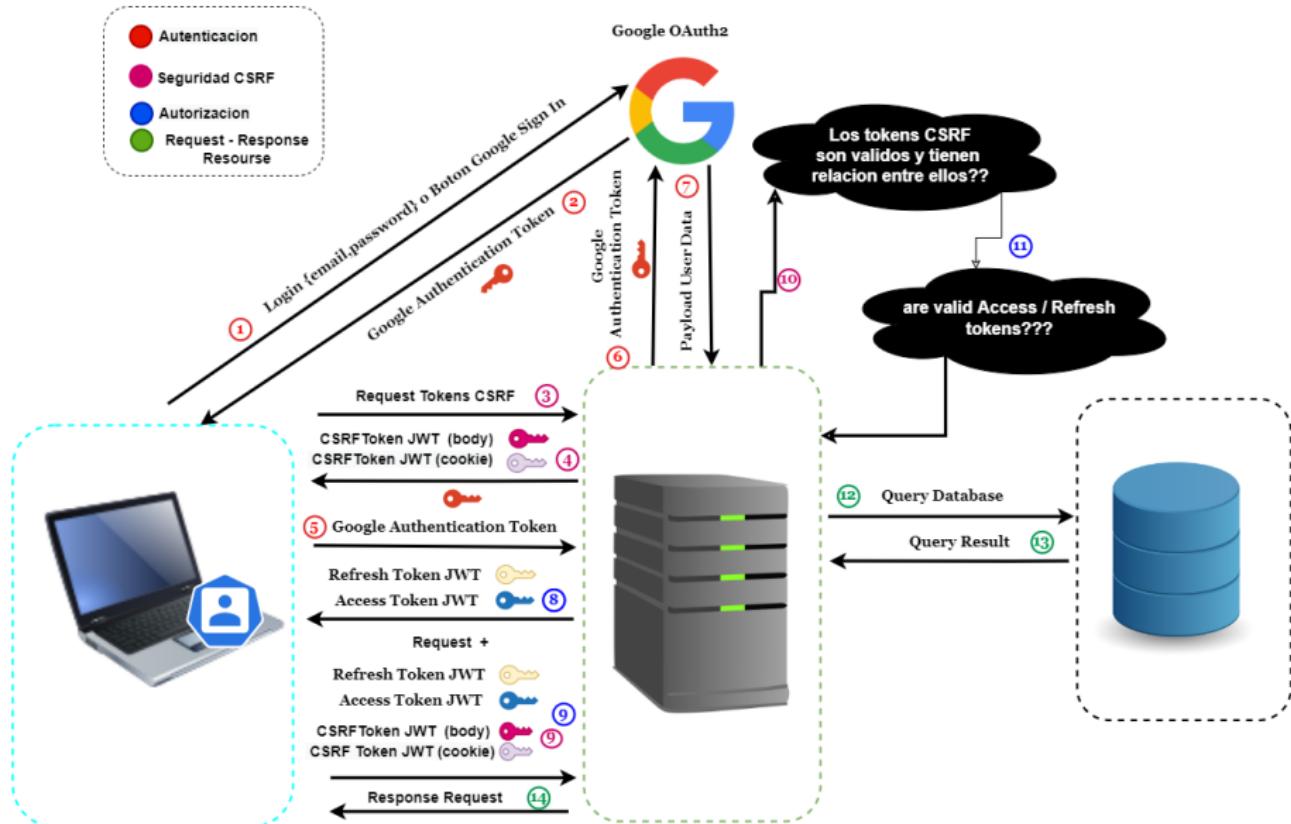
```
<!DOCTYPE html>
<html lang="en" data-redeviation-bs-uid="75716" class>
  <head>
    <meta charset="utf-8">
    <link rel="icon" href="/favicon.ico">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <meta name="theme-color" content="#000000">
    .. <meta name="csrf-token" content="TNBKXQzG-oqs9z_U8sqQ56JdPygn8aa6E1wg">
        == $0
```

Token guardado en tag html

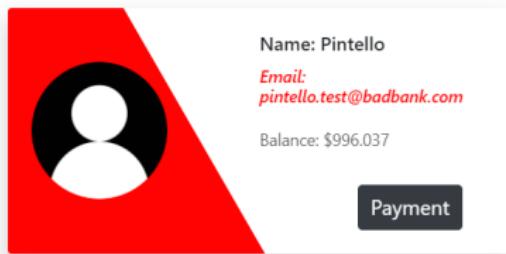
Name	Value	Domain	P.	Expires / Max-Age	▲	Size	HttpOnly	Secure	SameSite
token_refresh	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX..	mitbadbank.com	/	2022-04-21T22:32:38.925Z		185	✓	✓	Strict
_csrf-my-app	4_zfU5uGLo0Rjx6ezObf45ml	mitbadbank.com	/	2022-04-21T22:40:39.209Z		36	✓	✓	Strict

Token guardado den una cookie

# CS: Autenticacion, Autorizacion y CSRF



# Payment



Payment Account User X

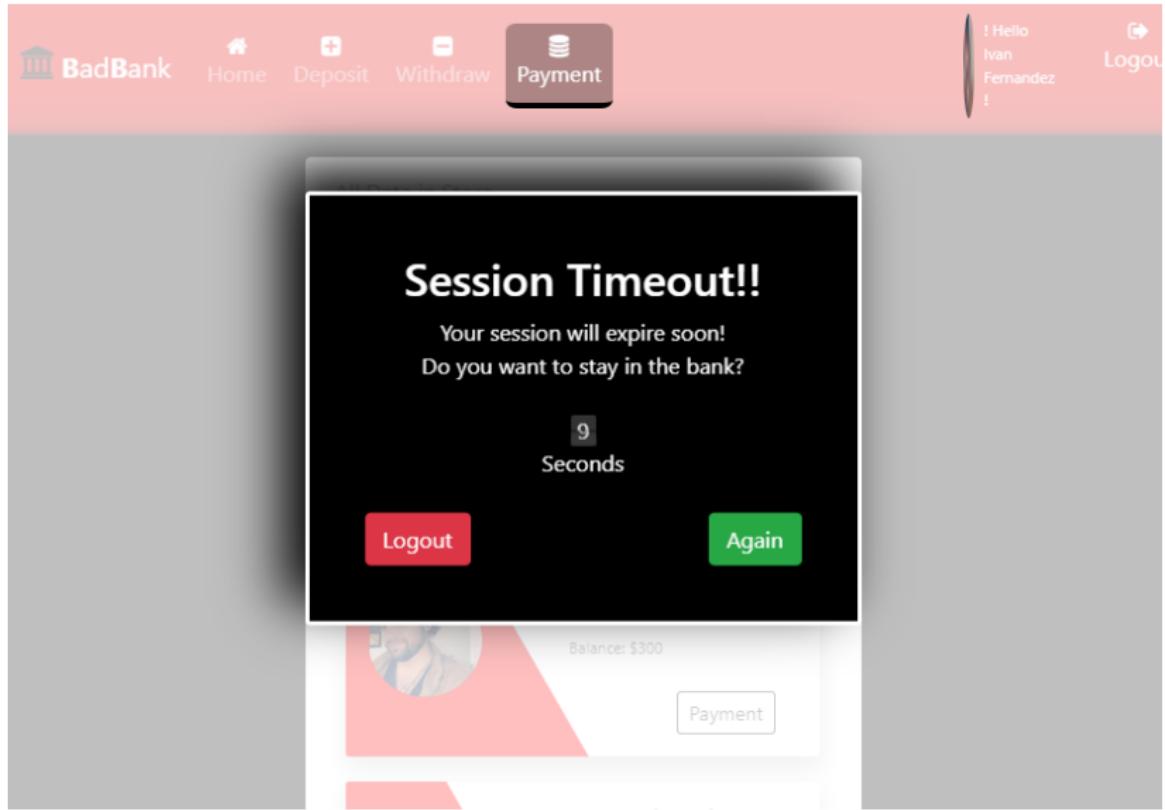
Email:

Amount:

Payment

Fields Validated

# Timeout Session



# Search Users



Massachusetts  
Institute of  
Technology

BadBank     Home     Deposit     Withdraw     Payment

Hello  
Ivan Fernandez !     Logout

All Data in Store

Search users by id, name or email

IVAN

Search...

Name: Ivan Fernandez  
Email: ivan.fernandez.g@usach.cl  
Balance: \$300

Payment

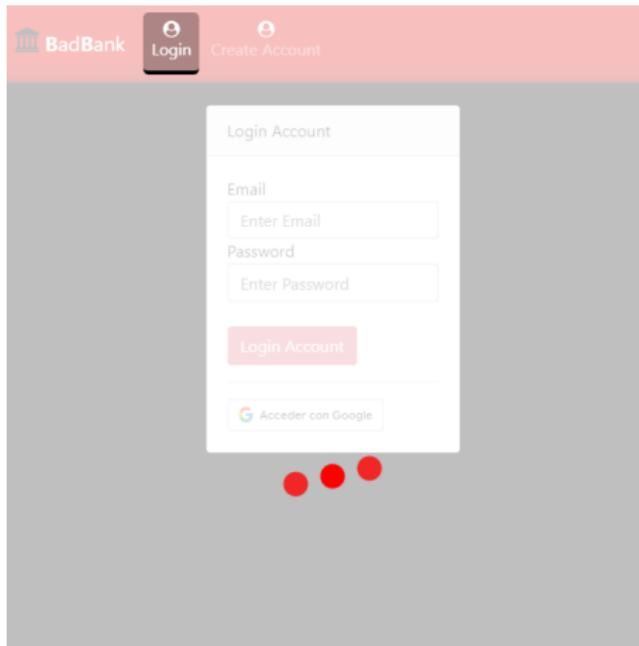
Name: Ivanincognito Fernandez  
Email: ivanincognitofernandez@gmail.com  
Balance: \$0

Payment

Name: Ivan  
Email: ivan.test@badbank.com  
Balance: \$6580

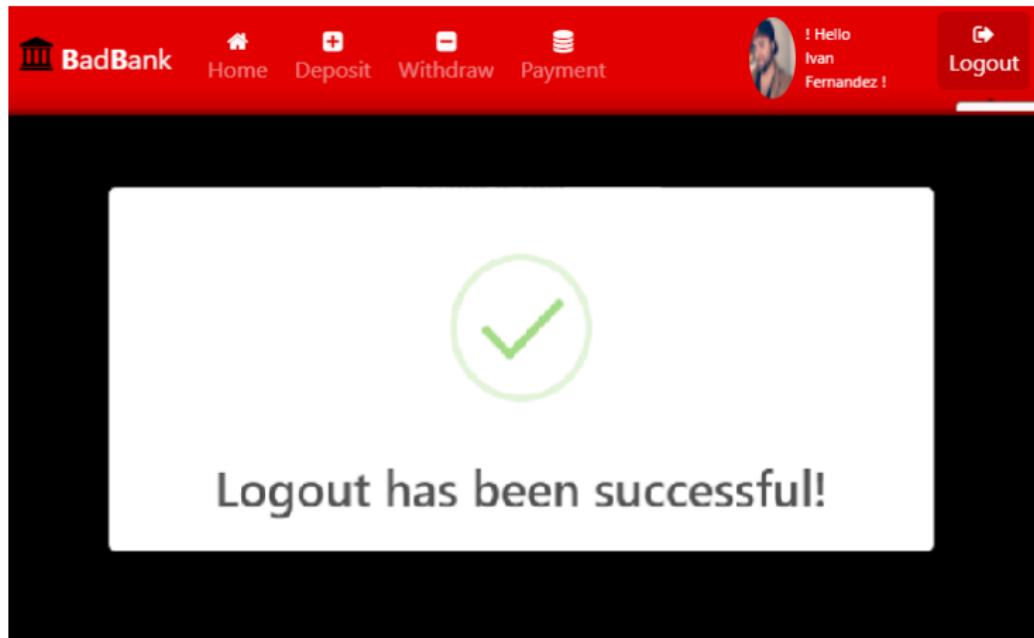
Payment

# Loading



Animación en la interfaz cuando se realizan **peticiones fetch al servidor**. Son tres círculos rojos girando sobre su eje central y crea un efecto de fondo blanco con opacidad

# Logout



La función logout hace una petición al servidor para **eliminar el Access Token y Refresh Token** de la base de datos Redis y manda una respuesta eliminando el Refresh token del navegador.

# Roles Backend Server

```
_id: ObjectId("623b45f899d43940cc7f9787")
role: "DB_ADMIN"
__v: 0

_id: ObjectId("623b4d0500abd05348e36061")
role: "DEV"
__v: 0

_id: ObjectId("623b4d0500abd05348e36062")
role: "USER"
__v: 0
```

3 Roles ("DBADMIN", "DEV", "USER") ingresados manualmente por seguridad a base de datos Mongo Atlas por medio de Mongo Compass. Estos roles son para **restringir los endpoint** a ciertos usuarios. Esto se implementa por medio de un middleware en los router de el server backend

# Contenido

- 1 Introducción y Objetivos
- 2 Three-Tiered Applications
- 3 Arquitectura Frontend
- 4 Autenticación
- 5 Arquitectura Backend
- 6 Funcionalidades de la Aplicación
- 7 Implementacion del Proyecto
- 8 Continous Integration Delivery CI/CD
- 9 Extras: Cyber Security, UI, Roles
- 10 Referencias

# Referencias

URL de la pagina web

<https://mitbadbank.com/>

URL del Repositorio Github

<https://github.com/IvanFernandezGracia/MIT-BadBankApp>

Linkedin

<https://www.linkedin.com/in/ivan-fg/>