

# Kata - FullStack

Apertura de cuentas para clientes nuevos

# Flujo Funcional

iPhone 12 Pro 390x844

Nuevo Cliente

Tipo de Documento  
Selección tipo de documento

Número de Documento \* (Mínimo 10 caracteres)  
1928837432

Nombre Completo (Mínimo 3 caracteres)  
John Doe

Email \*  
john.doe@example.com

Limpiar Agregar

Clientes Registrados

Documento	Nombre	Email	Fecha de Registro	Acciones
cc 1032449333	Diana D...	dd@email.co...	12/02/2026	<input type="button" value="Cuentas"/>
cc 1012389544	John Doe	john.doe@example.com	11/02/2026	<input type="button" value="Cuentas"/>

iPad 768x1024

Nuevo Cliente

Tipo de Documento  
Selección tipo de documento

Número de Documento \* (Mínimo 10 caracteres)  
1928837432

Nombre Completo (Mínimo 3 caracteres)  
John Doe

Email \*  
john.doe@example.com

Limpiar Agregar

Clientes Registrados

Documento	Nombre	Email	Fecha de Registro	Acciones
cc 1032449333	Diana D...	dd@email.co...	12/02/2026	<input type="button" value="Cuentas"/>
cc 1012389544	John Doe	john.doe@example.com	11/02/2026	<input type="button" value="Cuentas"/>

MacBook Pro 1440x900

CorpBank

Clientes Cuentas

Nuevo Cliente

Tipo de Documento  
Selección tipo de documento

Número de Documento \* (Mínimo 10 caracteres)  
1928837432

Nombre Completo (Mínimo 3 caracteres)  
John Doe

Email \*  
john.doe@example.com

Limpiar Agregar

Clientes Registrados

Documento	Nombre	Email	Fecha de Registro	Acciones
cc 1032449333	Diana D...	dd@email.com	12/02/2026	<input type="button" value="Cuentas"/>
cc 1012389544	John Doe	john.doe@example.com	11/02/2026	<input type="button" value="Cuentas"/>

Total de registros: 2 Mostrar: 5 << < 1 > >>

# FrontEnd

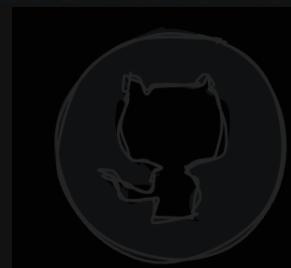
//localhost:4200

<https://fs-frontend-production.up.railway.app/customers>

# BackEnd

//localhost:8080

<https://fs-backend-production-43d9.up.railway.app/api/customers>



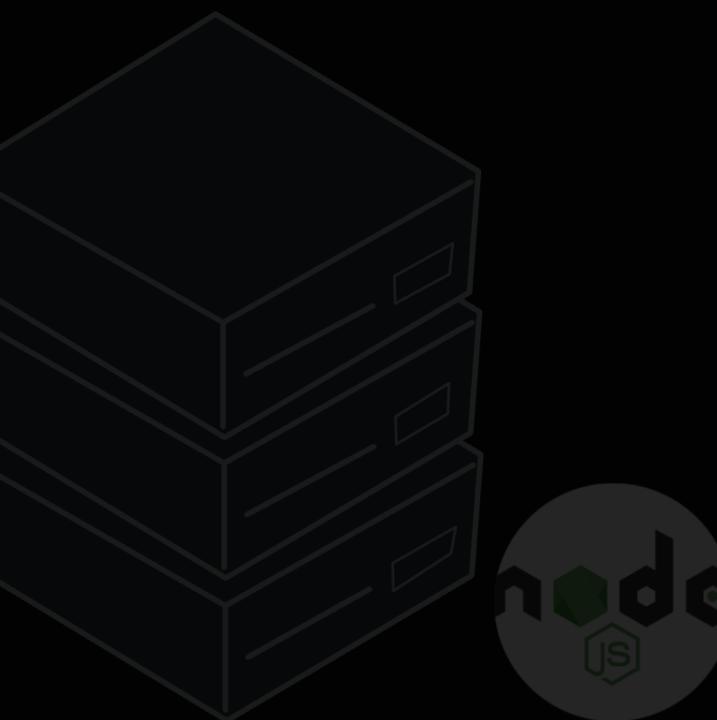
Monorepo



# DataBase

## FrontEnd

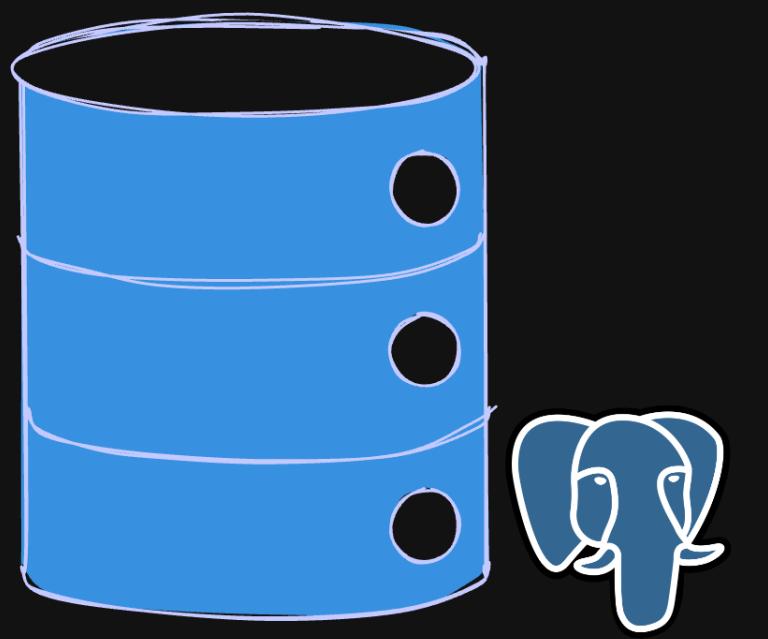
Angular V20  
Rxjs  
TypeScript  
Bootstrap  
ng-bootstrap  
ngx-toastr  
ng-icons  
ngx-translate



## BackEnd

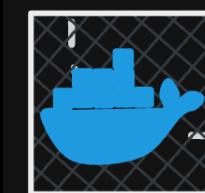
NodeJS  
Express  
Prisma  
cors  
pg  
typescript  
REST API  
(get, post, patch)  
i18n

Prisma  
ORM



## DataBase SQL

PostgreSQL  
DBeaver  
Function (account)  
Trigger



Docker



Railway  
PAAS



# Docker

## Creación DB

```
● docker-compose.yml
1 services:
2   db-service-stefanini:
3     image: 'postgres:latest'
4     container_name: postgresql_db_stefanini
5     restart: always
6
7   environment:
8     POSTGRES_USER: myUser
9     POSTGRES_PASSWORD: myPassword
10    POSTGRES_DB: myDataBase
11
12   volumes:
13     - postgres-data:/var/lib/postgresql
14
15   ports:
16     - "5433:5432"
17
18   shm_size: 128mb
19
20   volumes:
21     - postgres-data:
```

Configuración de contenedor  
PostgreSQL

The screenshot shows the Docker Desktop application interface. On the left, there's a sidebar with various icons for managing Docker resources. The main area is titled "Containers" and displays two running containers:

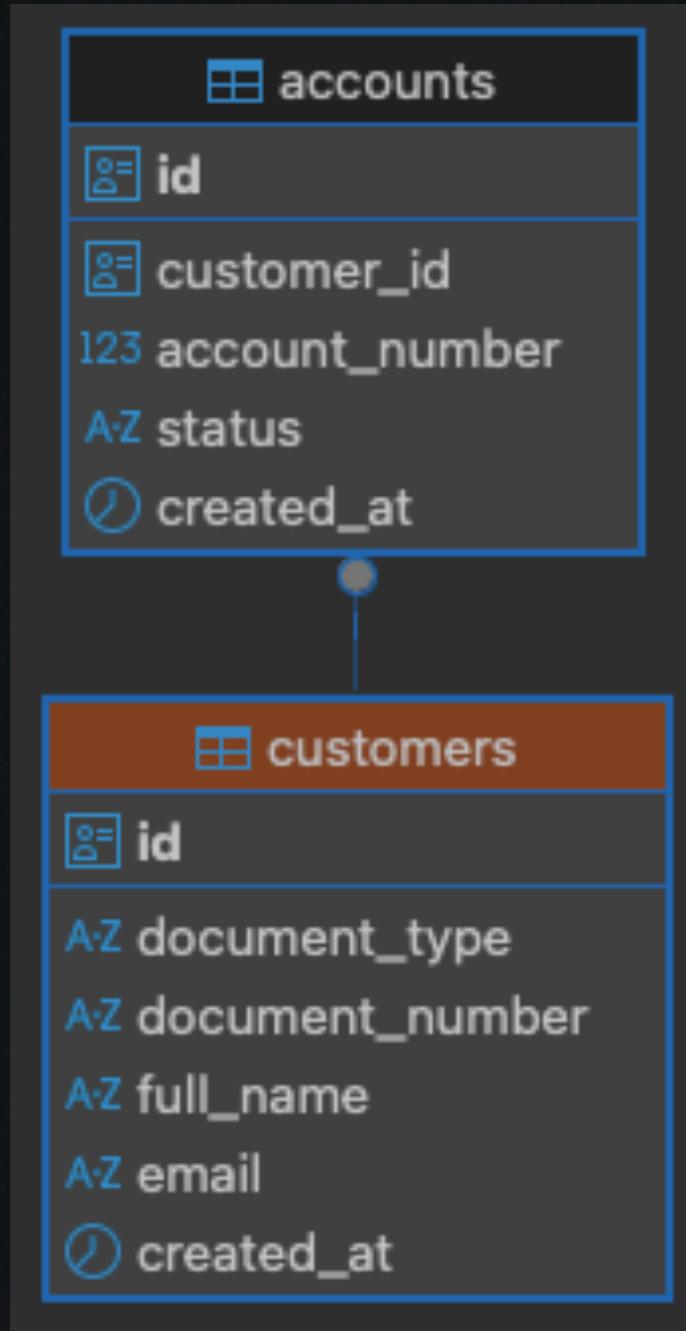
Name	Image	Port(s)	CPU (%)	Actions
data	.....	-	0.01%	[Edit] [More] [Delete]
postgresql_db_	postgres:latest	5433:5432	0.01%	[Edit] [More] [Delete]

At the bottom of the Docker Desktop window, there are resource usage metrics: RAM 0.70 GB, CPU 0.10%, and Disk: 1.77 GB used (limit 223.63 GB). There are also links for "Update available" and "Showing 2 items".

docker-compose up -d

# DataBase

## SQL y Relaciones



Relaciones

```
CREATE TABLE accounts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    customer_id UUID NOT NULL,
    account_number INTEGER UNIQUE,
    status VARCHAR(10) DEFAULT 'ACTIVE' CHECK (status IN ('ACTIVE', 'INACTIVE')),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    -- Regla de negocio: No permite crear cuenta si el cliente no existe
    CONSTRAINT fk_customer FOREIGN KEY(customer_id)
        REFERENCES customers(id)
        ON DELETE CASCADE
);
```

```
CREATE TABLE customers (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    document_type VARCHAR(3) NOT NULL CHECK (document_type IN ('CC', 'CE', 'PAS')),
    document_number VARCHAR(20) UNIQUE NOT NULL,
    full_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

Query para tablas

Función

```
CREATE OR REPLACE FUNCTION generate_unique_account_number()
RETURNS trigger AS $$
DECLARE
    new_acc_number INTEGER;
    exists_acc BOOLEAN;
BEGIN
    LOOP
        -- Generar un número aleatorio de 7 dígitos
        new_acc_number := floor(random() * (9999999 - 1000000 + 1) + 1000000);
        -- Verificar si ya existe en la tabla
        SELECT EXISTS(SELECT 1 FROM accounts WHERE account_number = new_acc_number) INTO exists_acc;
        -- Si no existe, salir del bucle
        EXIT WHEN NOT exists_acc;
    END LOOP;
    -- Asignar el número generado al nuevo registro
    NEW.account_number := new_acc_number;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_assign_account_number
BEFORE INSERT ON accounts
FOR EACH ROW
EXECUTE FUNCTION generate_unique_account_number();
```

Trigger



Monorepo

# BackEnd

## FrontEnd

Angular V20  
Rxjs  
TypeScript  
Bootstrap  
ng-bootstrap  
ngx-toastr  
ng-icons  
ngx-translate



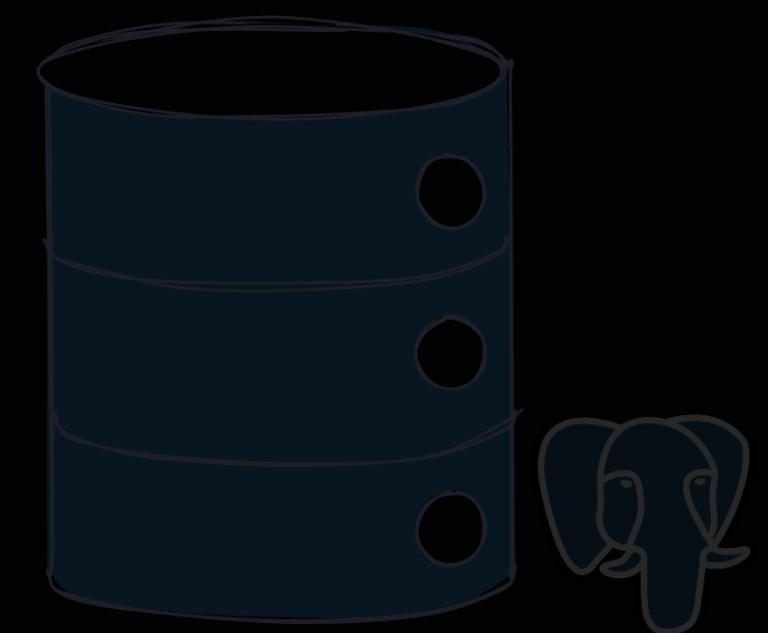
Railway  
PAAS



## BackEnd

NodeJS  
Express  
Prisma  
cors  
pg  
typescript  
REST API  
(get, post, patch)  
i18n

Prisma  
ORM



## DataBase SQL

PostgreSQL  
DBeaver  
Function (account)  
Trigger



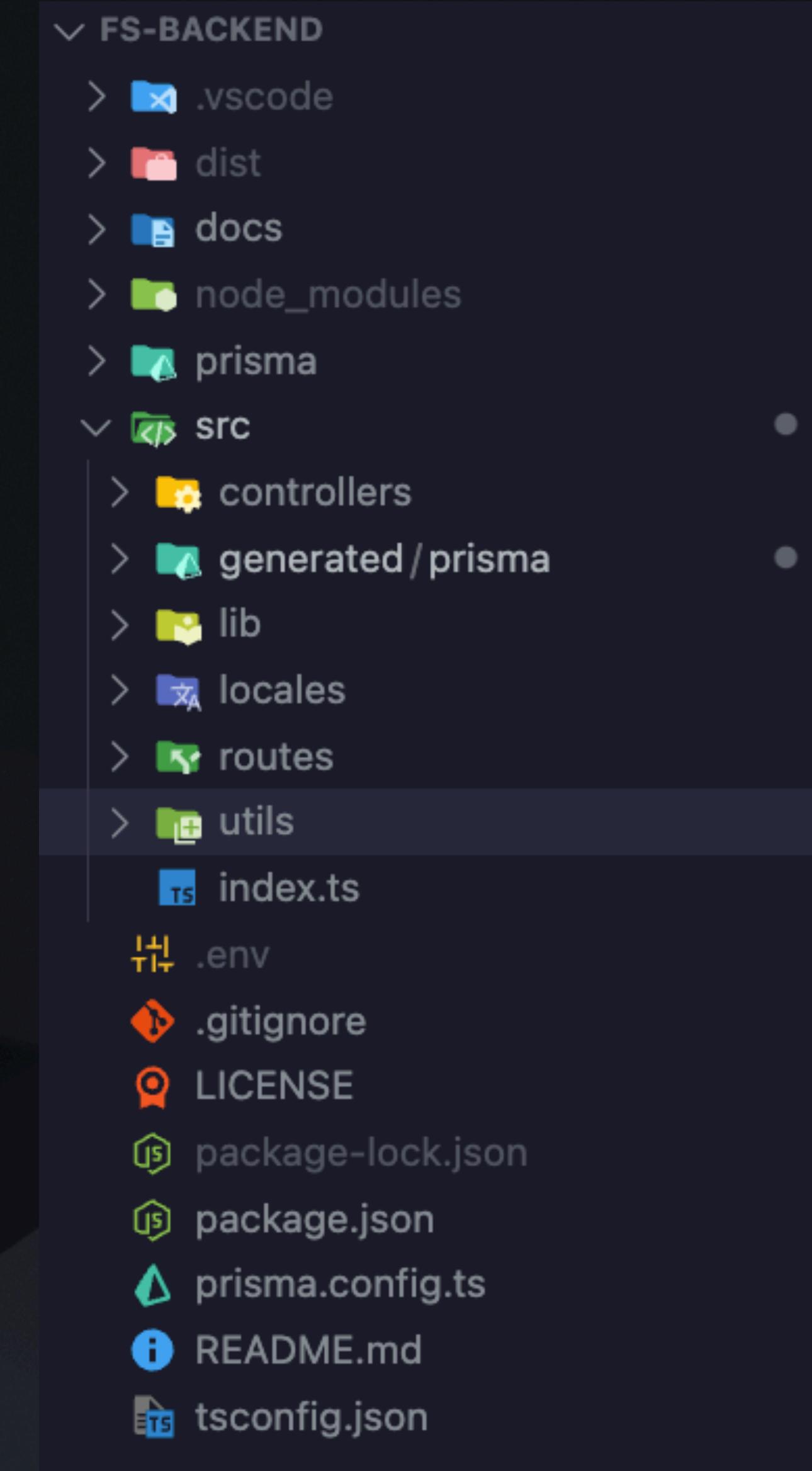
Docker

# BackEnd

## Configuración inicial

```
28 "dependencies": {  
29   "@prisma/adapter-pg": "^7.3.0",  
30   "@prisma/client": "^7.3.0",  
31   "cors": "^2.8.6",  
32   "dotenv": "^17.2.4",  
33   "express": "^5.2.1",  
34   "pg": "^8.18.0"  
35 },  
36 "devDependencies": {  
37   "@types/cors": "^2.8.19",  
38   "@types/express": "^5.0.6",  
39   "@types/node": "^25.2.2",  
40   "@types/pg": "^8.16.0",  
41   "nodemon": "^3.1.11",  
42   "prisma": "^7.3.0",  
43   "ts-node": "^10.9.2",  
44   "tsx": "^4.21.0",  
45   "typescript": "^5.9.3"  
46 }
```

Dependencias y versiones



Estructura de archivos



# BackEnd

## Prisma: Configuración y generación



```
prisma.config.ts
```

```
// This file was generated by Prisma, and assumes
// npm install --save-dev prisma dotenv
import "dotenv/config";
import { defineConfig } from "prisma/config";

export default defineConfig({
  schema: "prisma/schema.prisma",
  migrations: {
    path: "prisma/migrations",
    seed: "tsx prisma/seed.ts"
  },
  datasource: {
    url: process.env["DATABASE_URL"],
  },
});
```

Configuración Prisma

```
schema.prisma
```

```
prisma > schema.prisma > ...
Generate
generator client {
  provider = "prisma-client"
  output   = "../src/generated/prisma"
}

datasource db {
  provider = "postgresql"
}

/// This table contains check constraints and requires additional setup for migrations.
model accounts {
  id          String    @id @default(dbgenerated("gen_random_uuid()")) @db.Uuid
  customer_id String    @db.Uuid
  account_number Int      @unique
  status       String?   @default("ACTIVE") @db.VarChar(10)
  created_at  DateTime? @default(now()) @db.Timestamptz(6)
  customers   customers @relation(fields: [customer_id], references: [id], onDelete: cascade)
}

/// This table contains check constraints and requires additional setup for migrations.
model customers {
  id          String    @id @default(dbgenerated("gen_random_uuid()")) @db.Uuid
  document_type String    @db.VarChar(3)
  document_number String   @unique @db.VarChar(20)
  full_name    String    @db.VarChar(100)
  email        String    @unique @db.VarChar(100)
  created_at  DateTime? @default(now()) @db.Timestamptz(6)
  accounts     accounts[]
}
```

Datos de DB mapeados

Variable de connexion con DB

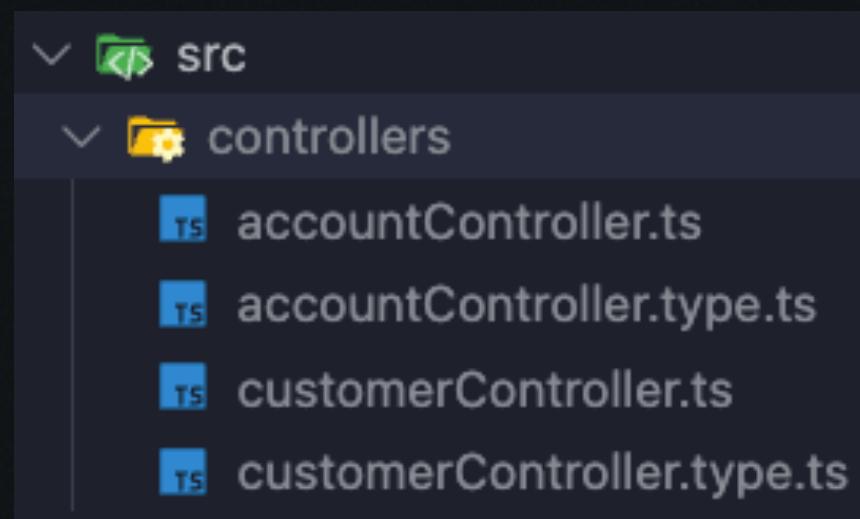
```
22 DATABASE_URL="postgresql://myUser:myPassword@localhost:5433/myDataBase?schema=public"
```

- generated/prisma
- internal
- models
- accounts.ts
- customers.ts
- browser.ts
- client.ts
- commonInputTypes.ts
- enums.ts
- models.ts

Modelos y funciones generadas

# BackEnd

Express: REST API's



Controladores

```
11 app.use(cors());
12 app.use(express.json());
13
14 // Routes
15 app.use('/api/customers', customerRoutes);
16 app.use('/api/accounts', accountRoutes);
17
18 const PORT = process.env.PORT ? Number(process.env.PORT) : 3000;
19 app.listen(PORT, () => {
20   console.log(`⚡ Servidor corriendo en puerto ${PORT}`);
21 })
```

Endpoints para Cliente y Cuenta

```
4 const router: Router = Router();
5
6 router.get('/', getCustomers);
7 router.post('/', createCustomer);
8
9 export default router;
```

Métodos HTTP para Cliente

```
4 const router: Router = Router();
5
6 router.get('/', getAccounts);
7 router.post('/', createAccount);
8 router.patch('/:id/status', updateAccountStatus);
9
10 export default router;
```

Métodos HTTP para Cuenta

# BackEnd

## Internacionalización de textos



```
res.status(201).json({
  message: t("success.account_created"),
  data: newAccount,
});
} catch (error: any) {
  res.status(500).json({
    error: t("errors.internal_server_error"),
    message: error.message ?? t("errors.account_created_error"),
  });
}
```

Utilidad de traducción

```
{} es.json •
src > locales > {} es.json > {} errors
1  {
2    "errors": {
3      "customer_not_found": "El cliente con documento {{id}} no existe.",
4      "account_created_error": "Ocurrió un error al crear la cuenta bancaria.",
5      "account_updated_error": "Ocurrió un error al actualizar la cuenta bancaria.",
6      "account_fetch_error": "Ocurrió un error al obtener la lista de cuentas bancarias.",
7      "account_update_error": "Ocurrió un error al actualizar la cuenta bancaria.",
8      "invalid_status": "Valor de estado invalido"
9    },
10   "success": {
11     "customer_created": "Cliente registrado exitosamente.",
12     "account_created": "Cuenta bancaria creada y asociada correctamente."
13   }
14 }
```

Textos centralizados por idioma

# BackEnd



## Peticiones y respuestas

The screenshot shows the Postman application interface. At the top, there's a header with 'API Network', a search bar, and various status indicators. Below the header, a list of API endpoints is shown, including a highlighted POST request to `https://fs-backend-production-43d9.up.railway.app/api/customers`. The main workspace displays a POST request to the same endpoint. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "document_type": "CC",  
3   "document_number": "1012389544",  
4   "full_name": "John Doe",  
5   "email": "john.doe@example.com"  
6 }
```

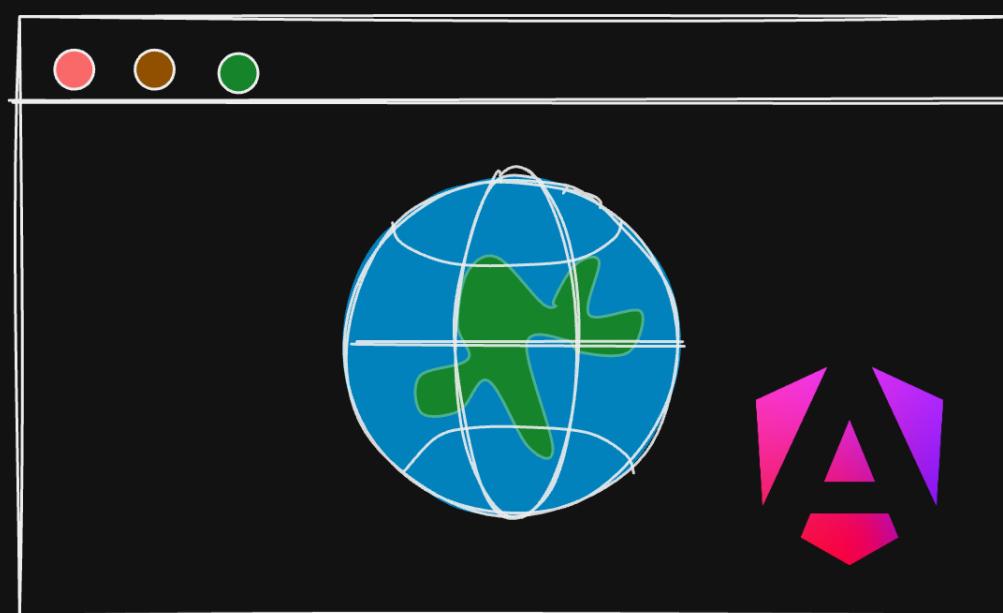
Below the body, the 'Test Results' tab shows a successful response with a status of `201 Created`, a duration of `485 ms`, and a size of `580 B`. The response body is displayed in JSON format:

```
1 {  
2   "message": "Cliente registrado exitosamente.",  
3   "data": {  
4     "id": "824922ea-1e30-4fc2-be11-677709c51eb3",  
5     "document_type": "CC",  
6     "document_number": "1012389544",  
7     "full_name": "John Doe",  
8     "email": "john.doe@example.com",  
9     "created_at": "2026-02-12T03:14:39.290Z"  
10    }  
11 }
```

At the bottom of the interface, there are navigation links for 'Terminal', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and help icons.



Monorepo



## FrontEnd

Angular V20  
Rxjs  
TypeScript  
Bootstrap  
ng-bootstrap  
ngx-toastr  
ng-icons  
ngx-translate



Railway  
PAAS



## BackEnd

NodeJS  
Express  
Prisma  
cors  
pg  
typescript  
REST API  
(get, post, patch)  
i18n

# FrontEnd



## DataBase SQL

PostgreSQL  
DBeaver  
Function (account)  
Trigger



Docker

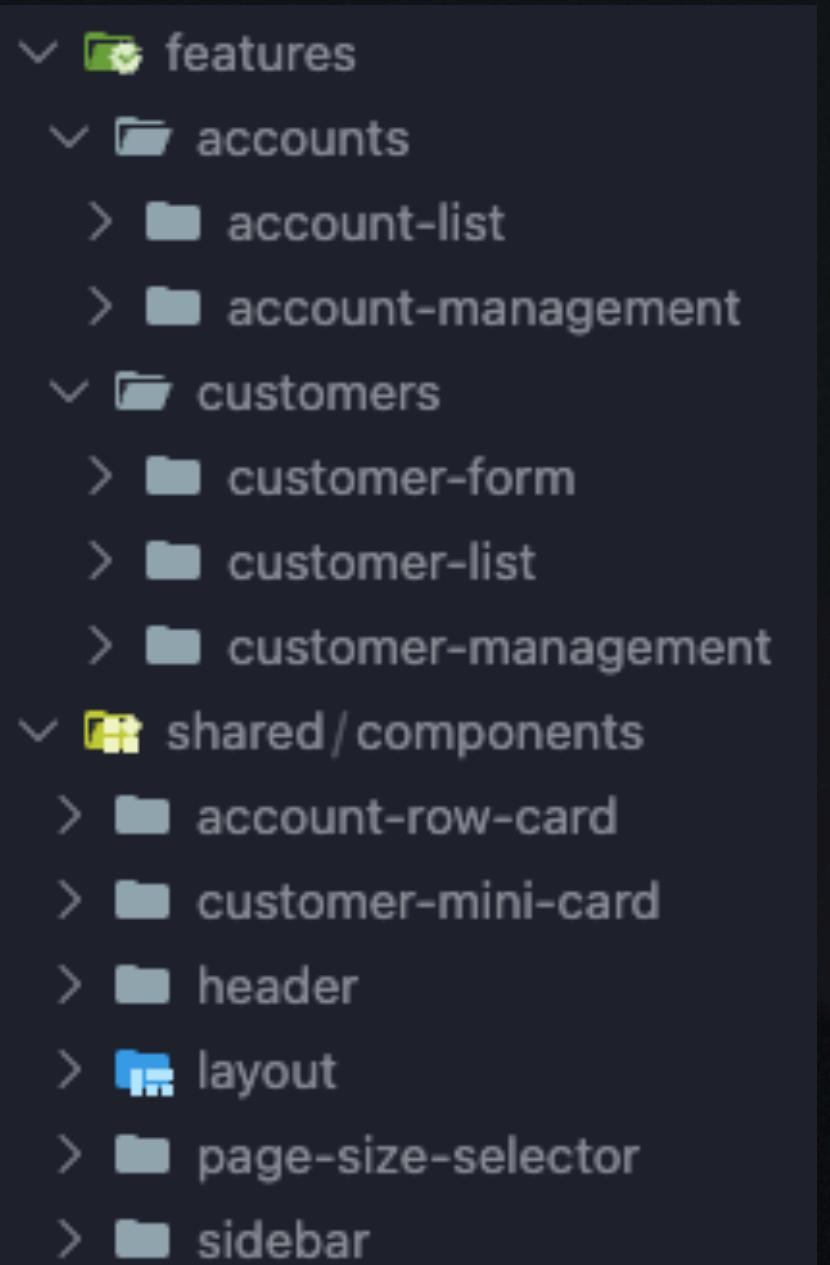
# FrontEnd



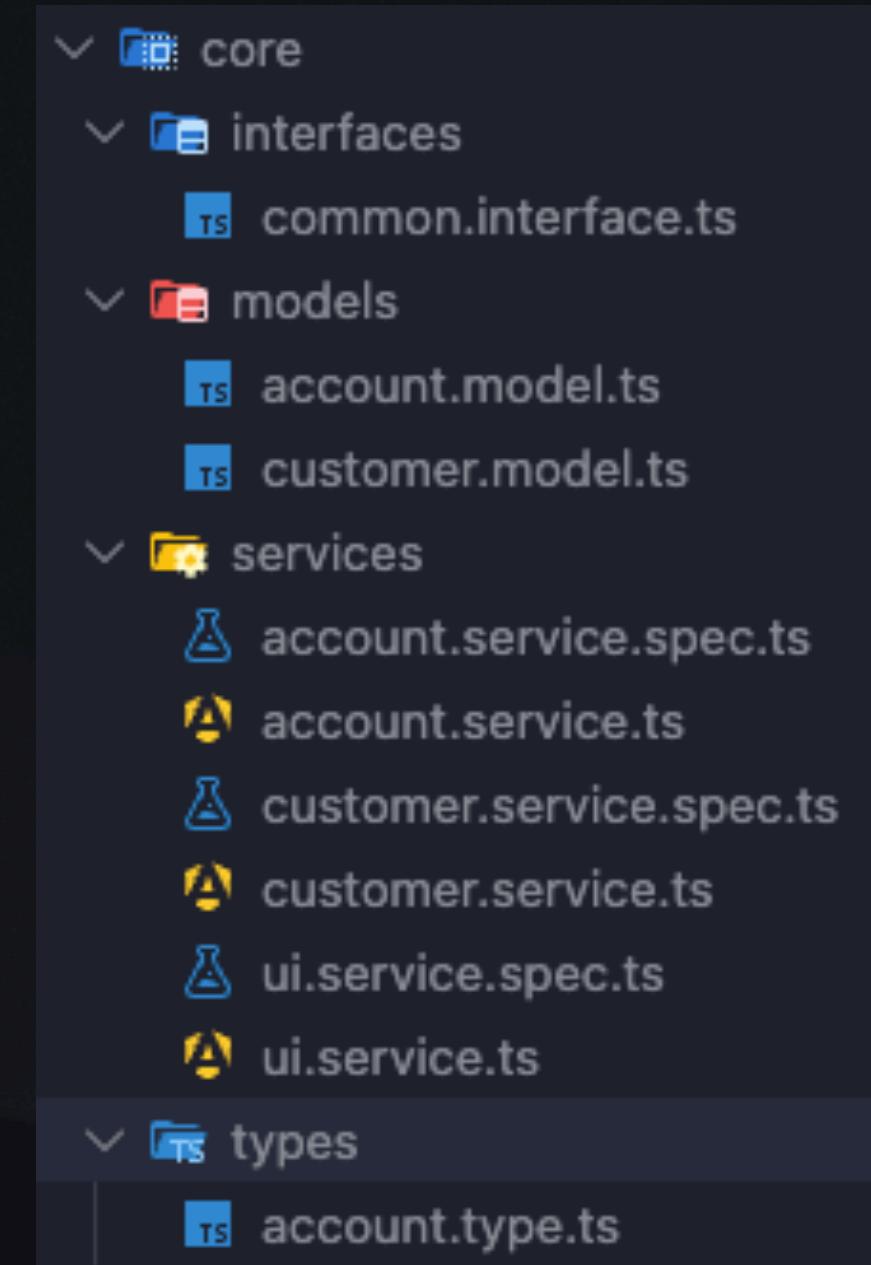
## Configuración inicial

```
25 "dependencies": {  
26   "@angular/animations": "^20.3.16",  
27   "@angular/cdk": "^20.2.14",  
28   "@angular/common": "^20.3.0",  
29   "@angular/compiler": "^20.3.0",  
30   "@angular/core": "^20.3.0",  
31   "@angular/forms": "^20.3.0",  
32   "@angular/platform-browser": "^20.3.0",  
33   "@angular/router": "^20.3.0",  
34   "@ng-bootstrap/ng-bootstrap": "^19.0.1",  
35   "@ng-icons/bootstrap-icons": "^33.1.0",  
36   "@ng-icons/core": "^32.5.0",  
37   "@ngx-translate/core": "^17.0.0",  
38   "@ngx-translate/http-loader": "^17.0.0",  
39   "bootstrap": "^5.3.6",  
40   "ngx-toastr": "^19.1.0",  
41   "rxjs": "~7.8.0",  
42   "tslib": "^2.3.0"  
43 },  
44 "devDependencies": {  
45   "@angular/build": "^20.3.15",  
46   "@angular/cli": "^20.3.15",  
47   "@angular/compiler-cli": "^20.3.0",  
48   "@angular/localize": "^20.3.16",  
49   "@types/jasmine": "^5.1.0",  
50   "jasmine-core": "^5.9.0",  
51   "karma": "~6.4.0",  
52   "karma-chrome-launcher": "~3.2.0",  
53   "karma-coverage": "~2.2.0",  
54   "karma-jasmine": "~5.1.0",  
55   "karma-jasmine-html-reporter": "~2.1.0",  
56   "typescript": "~5.9.2"  
57 }
```

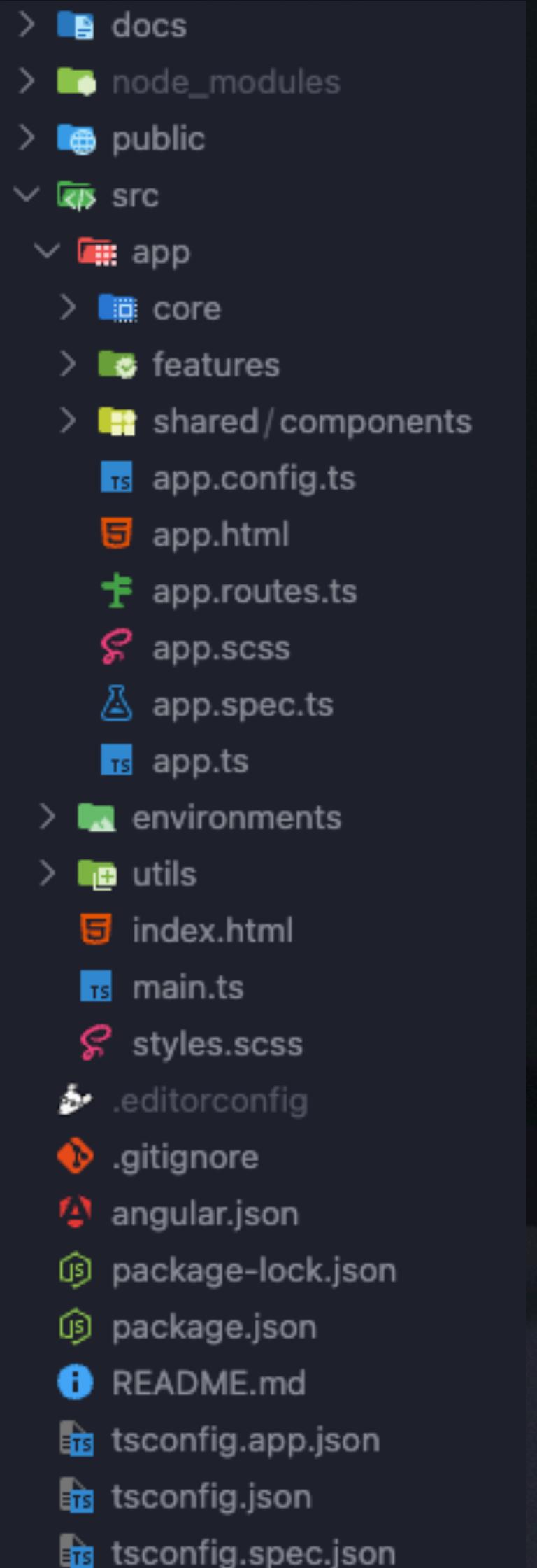
Dependencias y versiones



Componentes



Modelos, Tipos y Servicios



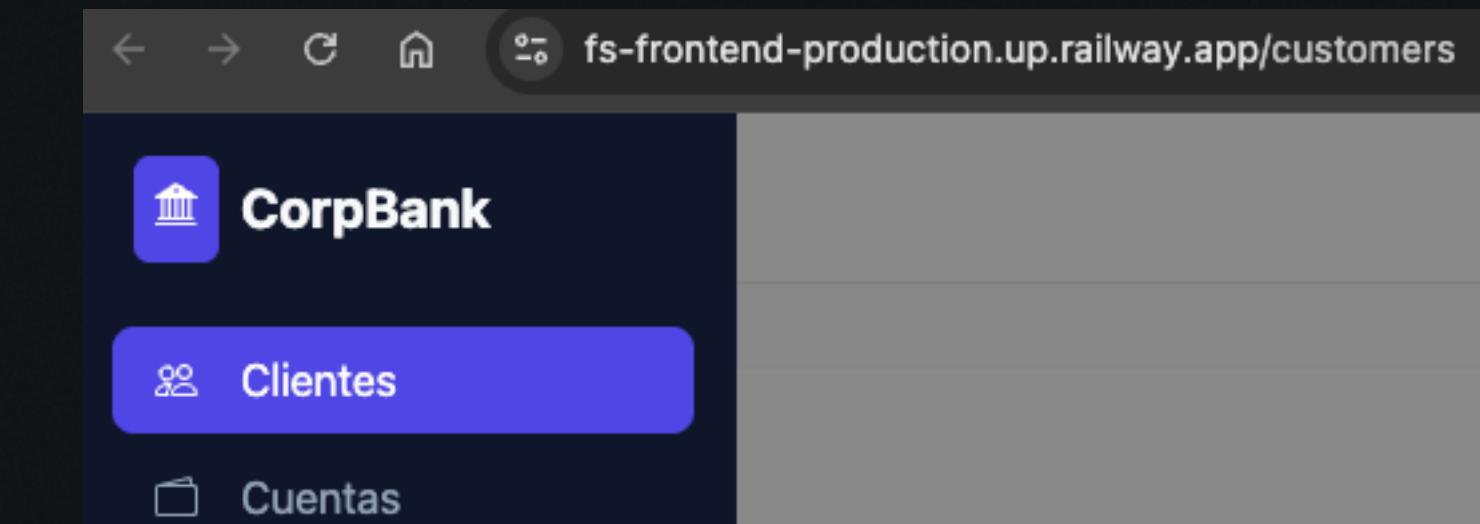
Estructura de archivos

# FrontEnd

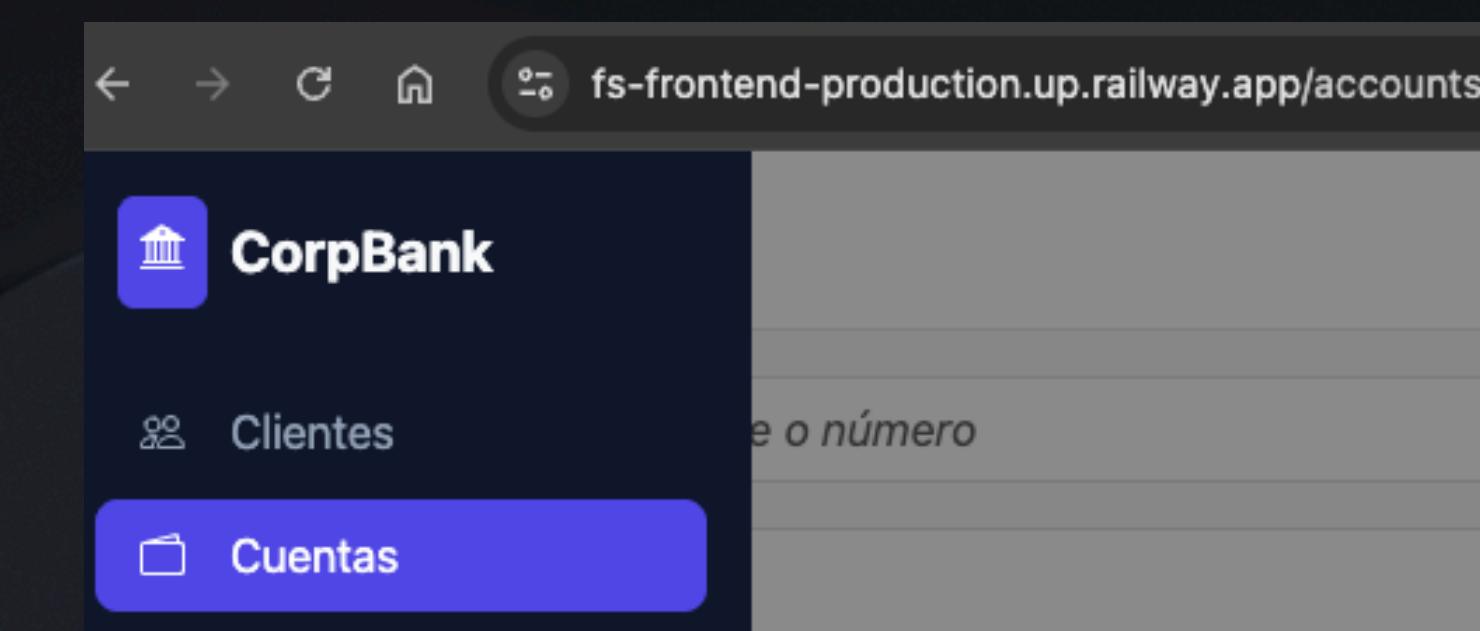


## Rutas

```
app.routes.ts M X  
src > app > app.routes.ts > ...  
1 import { Routes } from '@angular/router';  
2 import { Layout } from './shared/components/layout/layout';  
3  
4 export const routes: Routes = [  
5   {  
6     path: '',  
7     component: Layout,  
8     children: [  
9       { path: '', redirectTo: 'customers', pathMatch: 'full' },  
10      {  
11        path: 'customers',  
12        loadComponent: () =>  
13          import('./features/customers/customer-management/customer-management').then(  
14            (m) => m.CustomerManagement,  
15          ),  
16        },  
17        {  
18          path: 'accounts',  
19          loadComponent: () =>  
20            import('./features/accounts/account-management/account-management').then(  
21              (m) => m.AccountManagement,  
22            ),  
23        },  
24      ],  
25    },  
26    { path: '**', redirectTo: 'customers' },  
27  ];
```



Sección Clientes



Sección Cuentas

# FrontEnd



## Servicios

```
9  @Injectable({
10 |   providedIn: 'root',
11 | })
12 export class CustomerService {
13   private readonly http = inject(HttpClient);
14   private readonly apiUrl = `${environment.apiUrl}/customers`;
15
16   private readonly customerCreatedSource = new Subject<void>();
17   customerCreated$ = this.customerCreatedSource.asObservable();
18
19   readonly #customers = signal<Customer[]>([]);
20   customers = this.#customers.asReadOnly();
21
22   // GET customers
23 >   getCustomers(...)
24 }
25
26   // POST create customer
27 >   createCustomer(customer: Partial<Customer>): Observable<any> { ...
28 }
29 }
```

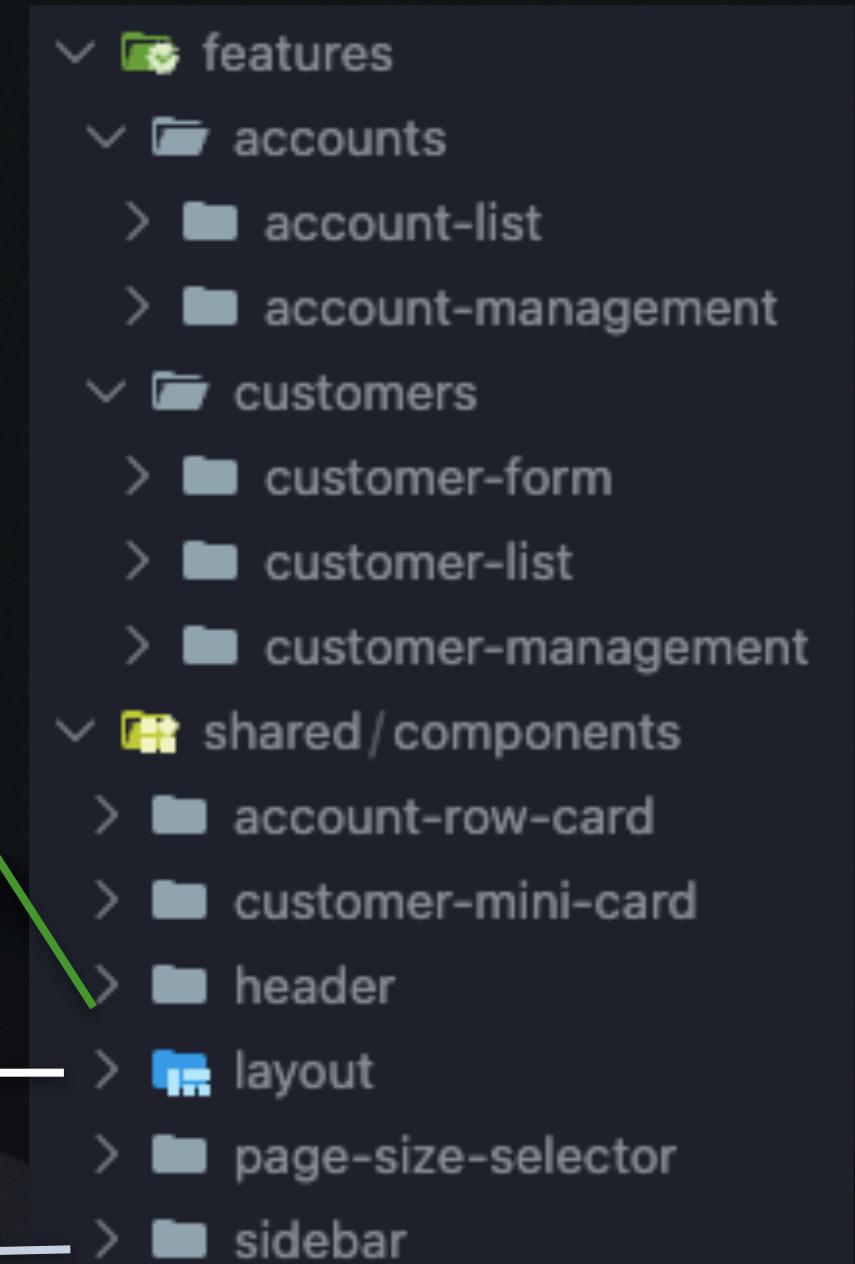
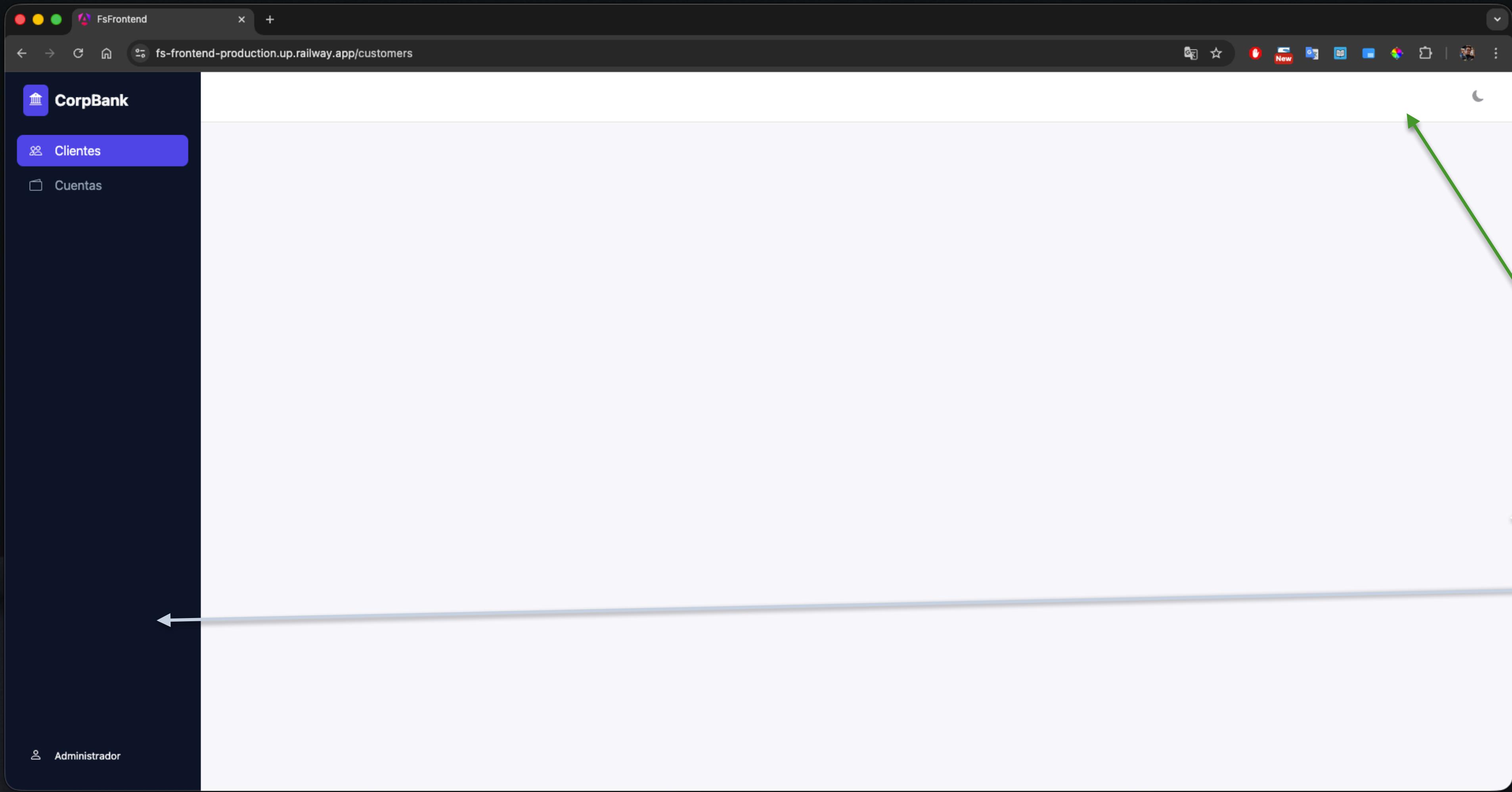
CustomerService

```
9  @Injectable({
10 |   providedIn: 'root',
11 | })
12 export class AccountService {
13   private readonly http = inject(HttpClient);
14   private readonly apiUrl = `${environment.apiUrl}/accounts`;
15
16   private readonly accountChangedSource = new Subject<void>();
17   accountChanged$ = this.accountChangedSource.asObservable();
18
19   readonly #accounts = signal<Account[]>([]);
20   accounts = this.#accounts.asReadOnly();
21
22   // GET accounts
23 >   getAccounts(...)
24 }
25
26   // POST create account
27 >   createAccount(documentNumber: string = ''): Observable<any> { ...
28 }
29
30   // PATCH status account
31 >   updateStatus(accountId: string, newStatus: TAccountStatus): Observable<any> { ...
32 }
33 }
```

AccountService

# FrontEnd

## Componentes



# FrontEnd

## Componentes



The screenshot shows the CorpBank Frontend application interface. The sidebar on the left has 'Cuentas' selected. The main area shows a 'Nuevo Cliente' form and a 'Clientes Registrados' table.

**Nuevo Cliente Form:**

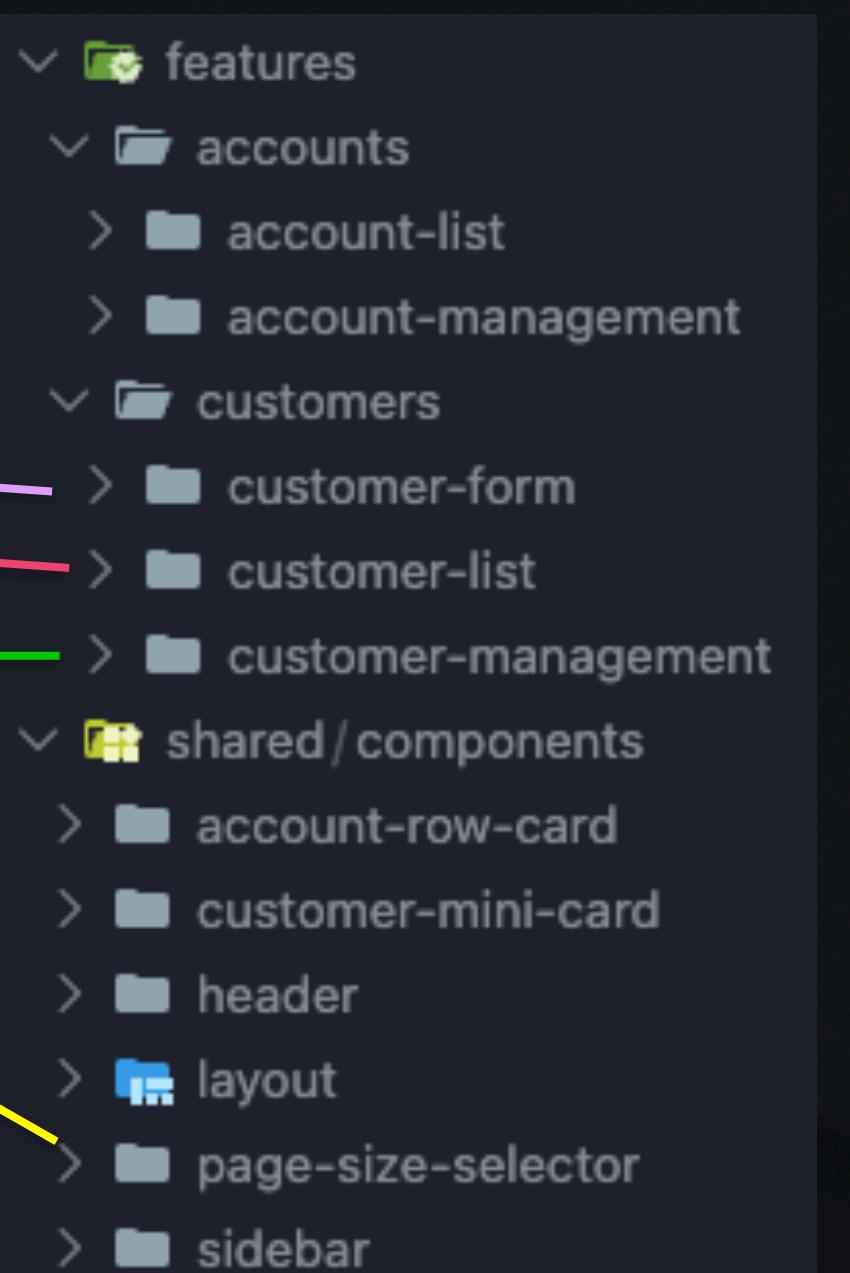
- Tipo de Documento: Seleccionar tipo de documento
- Número de Documento \* (Mínimo 10 caracteres): 1928837432
- Nombre Completo (Mínimo 3 caracteres): John Doe
- Email \*: john.doe@example.com

**Clientes Registrados Table:**

Documento	Nombre	Email	Fecha de Registro	Acciones
cc 1032449333	Diana Doe	dd@email.com	12/02/2026	<a href="#">Cuentas</a>
cc 1012389544	John Doe	john.doe@example.com	11/02/2026	<a href="#">Cuentas</a>

Total de registros: 2

Mostrar: 5



# FrontEnd

## Componentes



The screenshot illustrates the FrontEnd architecture, specifically the 'accounts' feature. The sidebar navigation includes 'Cuentas' (selected), 'Clientes', and 'Administrador'. The main content displays a list of accounts for 'Diana Doe' (CC: 1032449333) and 'John Doe' (CC: 1012389544). The UI components are mapped to the following tree structure:

- features/accounts:
  - account-list
  - account-management
- customers:
  - customer-form
  - customer-list
  - customer-management
- shared/components:
  - account-row-card
  - customer-mini-card
  - header
  - layout
  - page-size-selector
  - sidebar

Sección Cuentas

# FrontEnd



## Internacionalización de textos

The screenshot shows a code editor interface with two tabs open:

- es.json**: An i18n configuration file. It defines a structure for translating components like buttons, tables, and toast messages. For example, it maps "clear" to "Limpiar" and "add" to "Agregar".
- es.json**: A JSON file containing translations for various UI elements. It includes keys for buttons, tables, toast messages, sidebar components, customer forms, and lists. For instance, it defines "total\_record" as "Total de registros:" and "size\_label" as "Mostrar:".

The right side of the interface displays a hierarchical tree view of the JSON data, with nodes expanded to show their values. Labels like "success", "error", and "title" are also visible, indicating different message types.

At the bottom, there's an "Open Web Editor" button.

```
<div class="mb-4">
  <label class="form-label fw-medium" for="txt-email">
    {{ 'customer_form.email.label' | translate}}
  </label>
  <input
```

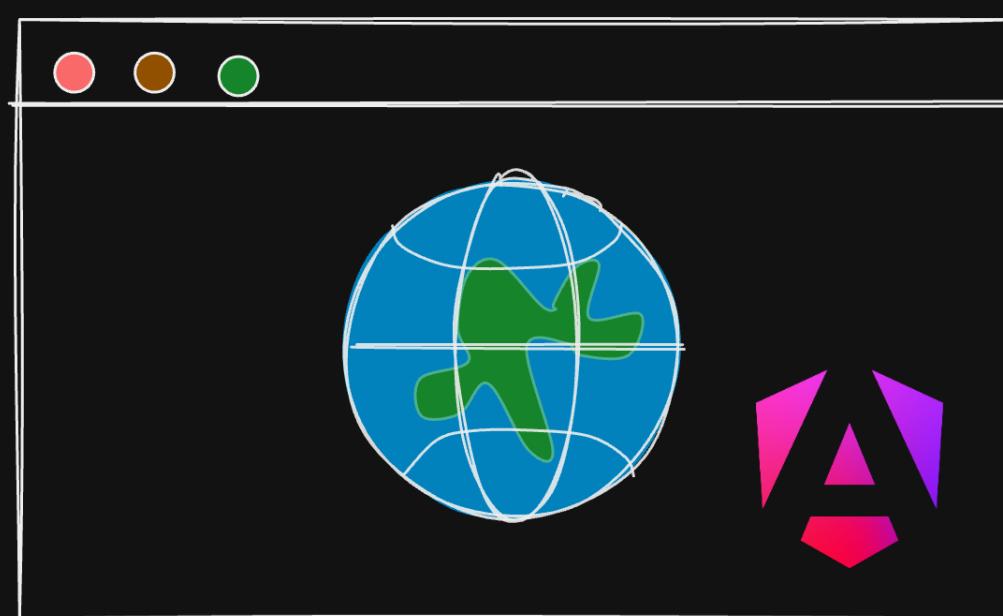
ng-translate

```
this.toastr.success(
  t('toast_msg.success.customer_register',
  { name: res.data.full_name }),
  t('toast_msg.success.title')
);
```

Utilidad de traducción



Monorepo



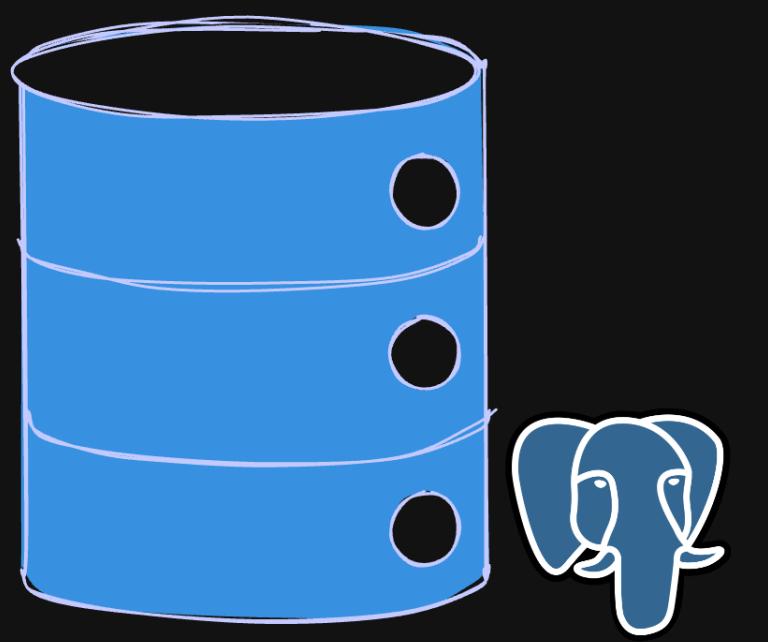
Angular V20  
Rxjs  
TypeScript  
Bootstrap  
ng-bootstrap  
ngx-toastr  
ng-icons  
ngx-translate



BackEnd

NodeJS  
Express  
Prisma  
cors  
pg  
typescript  
REST API  
(get, post, patch)  
i18n

Prisma  
ORM

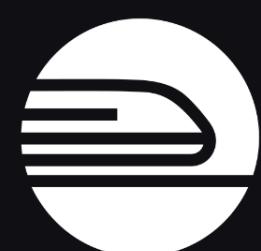


DataBase SQL

PostgreSQL  
DBeaver  
Function (account)  
Trigger



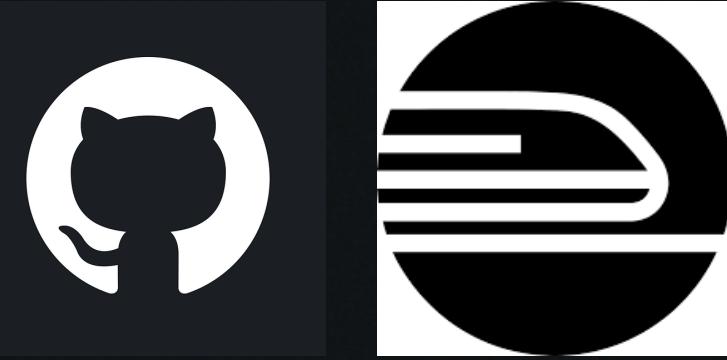
Docker



Railway  
PAAS

# GitHub y Railway

Control de versiones y Despliegue en la nube



Screenshot of the GitHub repository page for `jleon253 / fs-frontend`.

Description: Frontend ligero en Angular V20 para crear, leer y modificar clientes y cuentas bancarias.  
<https://fs-frontend-production.up.railway.app/customers>

Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings

Watch 0, Forks 0, Watching 0, Branch 1, Tags 0, Activity

Public repository

<https://github.com/jleon253/fs-frontend>

Screenshot of the GitHub repository page for `jleon253 / fs-backend`.

Description: Backend ligero en TypeScript (Express + Prisma) para gestionar clientes y cuentas bancarias.  
<https://fs-backend-production-43d9.up.railway.app/api/customers>

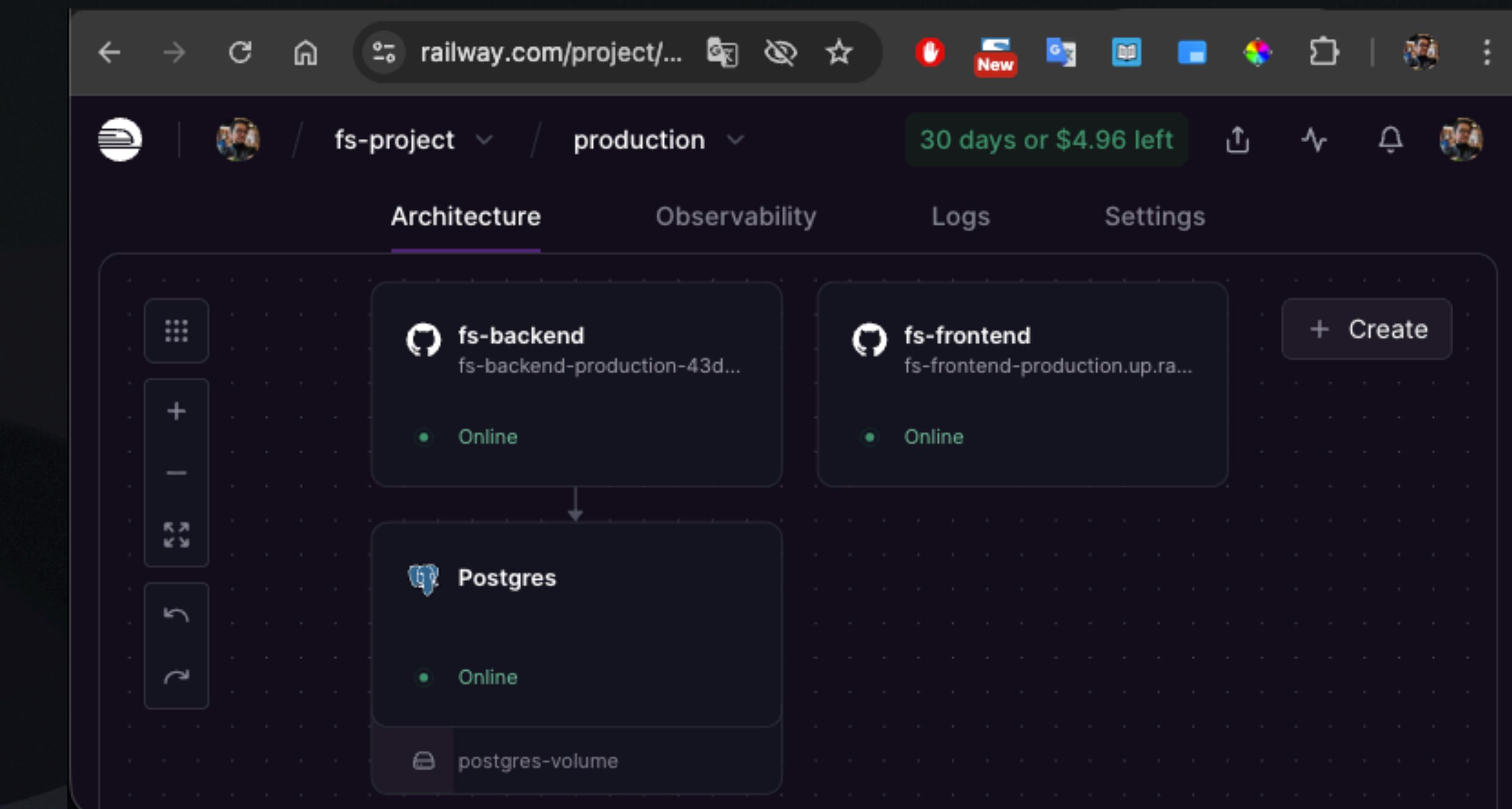
Apache-2.0 license

Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings

Watch 0, Forks 0, Watching 0, Branch 1, Tags 0, Activity

Public repository

<https://github.com/jleon253/fs-backend>



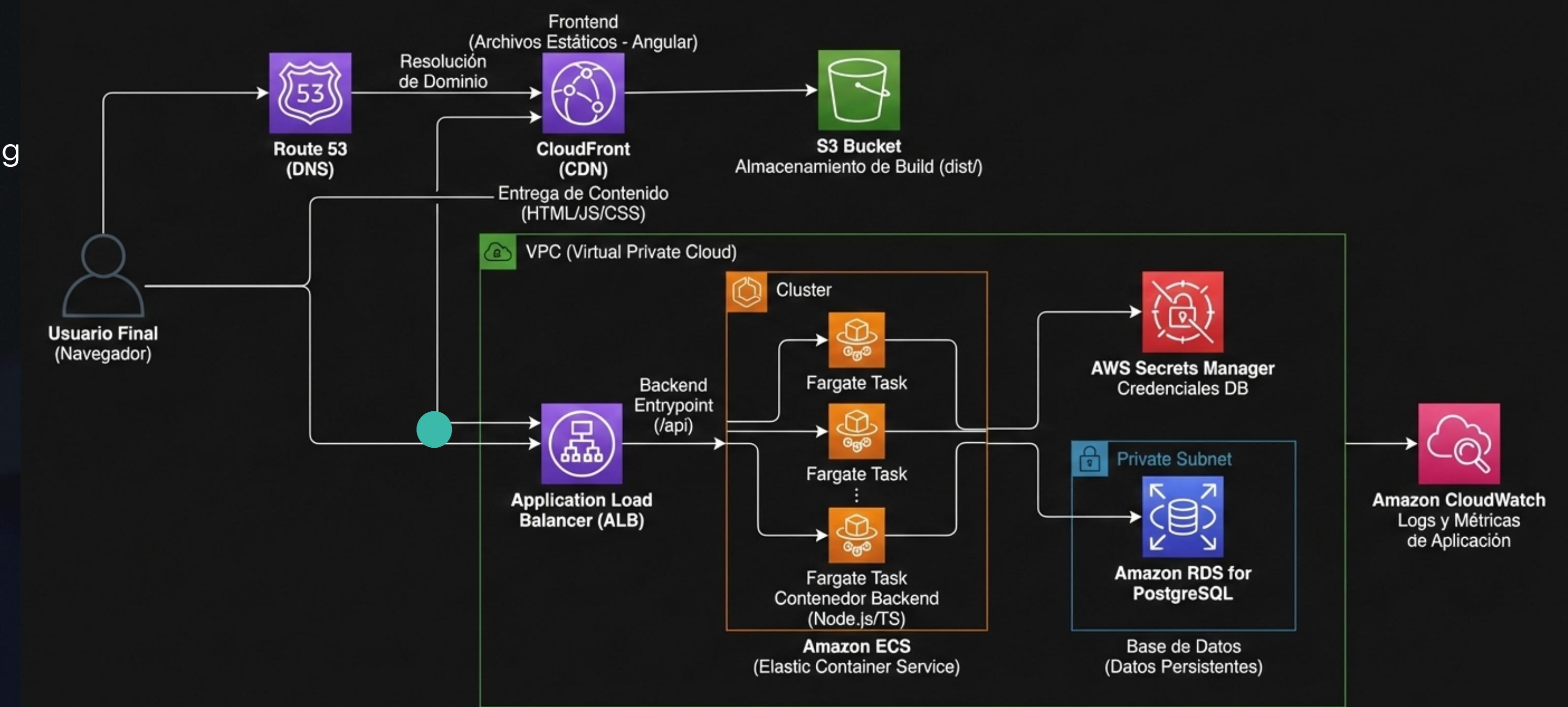
Despliegue: DB, Back y Front

# Respuestas sobre AWS

# AWS

## Respuestas

- Qué servicio usarías para la base de datos?
  - Usaría AWS RDS (Relational Data Service para PostgreSQL) con Aurora Serverless v2
- Cómo desplegarías el backend en AWS?
  - En AWS ECS con Fargate Serverless o ECS Auto Scaling Group
  - Se disponibiliza usando [apigateway](#)
  - Lambdas (si fueran pequeñas funciones)
- Dónde hospedarías el frontend?
  - En S3 (al ser SPA) con CloudFront o Amplify (si usara SSR)
  - S3 almacena archivos (imágenes, docs, etc)
  - Route53 para el DNS
  - ALB (Application Load Balancer) para repartir tráfico
  - AWS Secrets Manager para variables de entorno
- Cómo verías logs en producción?
  - Implementando CloudWatch en ambientes pre-PROD
  - Hacer pipelines de CloudWatch hacia OpenSearch / ElasticSearch en ambientes PROD
  - Proyectos bancarios (Prod a gran escala), generar métricas y almacenar solo logs de error
  - Monitoreo con Grafana
  - Analítica con Glue y Athena



**Gracias**