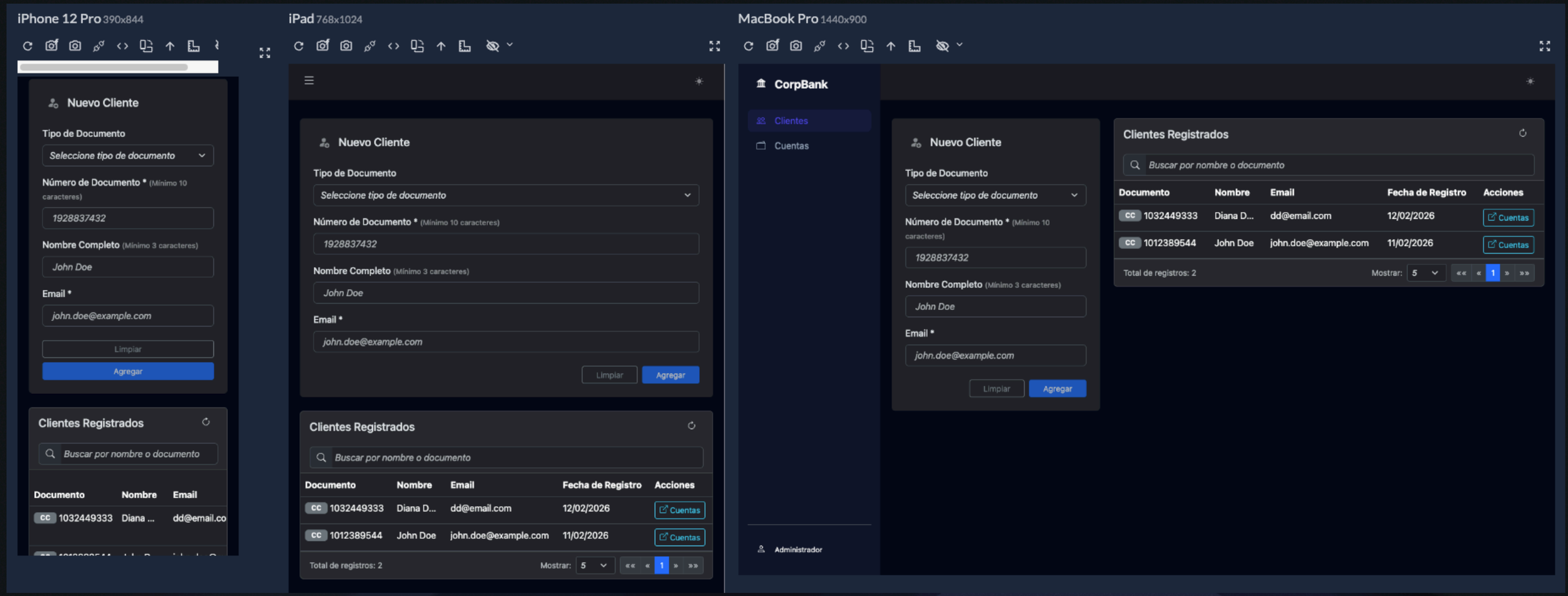
Kata - FullStack

Apertura de cuentas para clientes nuevos

John León - 12/Feb/2026

Flujo Funcional



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

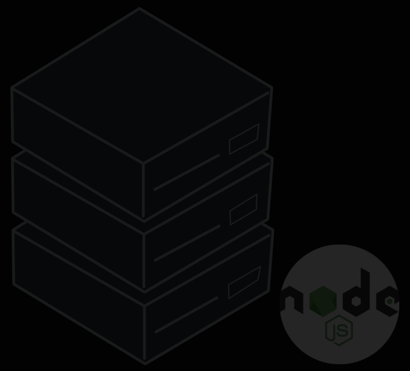
DataBase

FrontEnd

Angular V20
Rxjs
TypeScript

Bootstrap
ng-bootstrap
ngx-toastr
ng-icons

ngx-translate



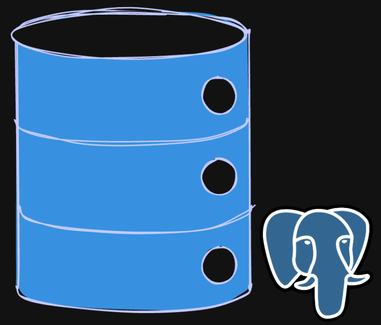
BackEnd

NodeJS
Express
Prisma

cors
pg
typescript

REST API
(get, post, patch)
i18n

Prisma
ORM



DataBase SQL

PostgreSQL

DBeaver

Function (account)
Trigger

Docker

Railway
PAAS

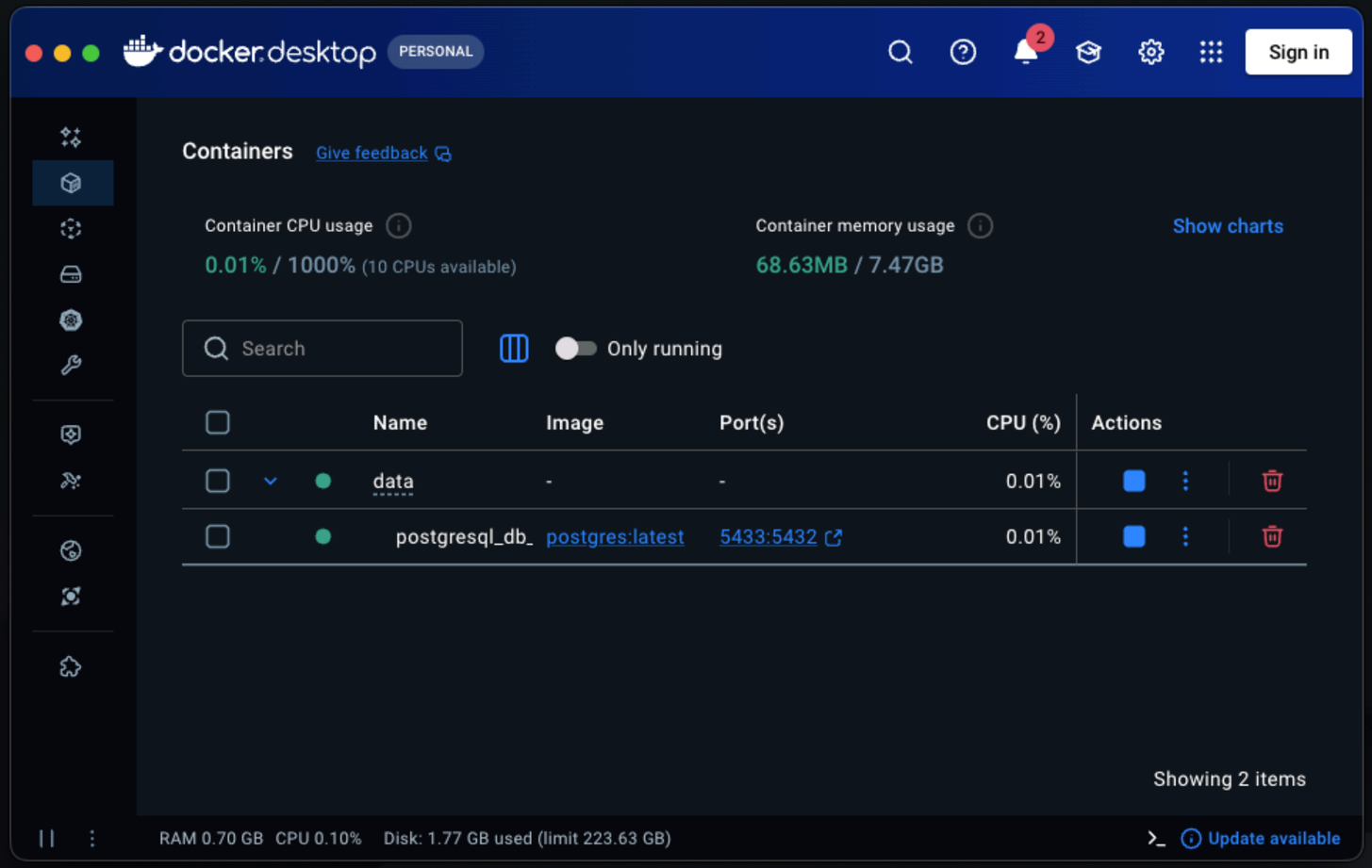
Docker

Creación DB

docker-compose.yml

```yaml
services:
  db-service-stefanini:
    image: 'postgres:latest'
    container_name: postgresql_db_stefanini
    restart: always

    environment:
      POSTGRES_USER: myUser
      POSTGRES_PASSWORD: myPassword
      POSTGRES_DB: myDataBase

    volumes:
      - postgres-data:/var/lib/postgresql

    ports:
      - "5433:5432"

    shm_size: 128mb

volumes:
    postgres-data:
```

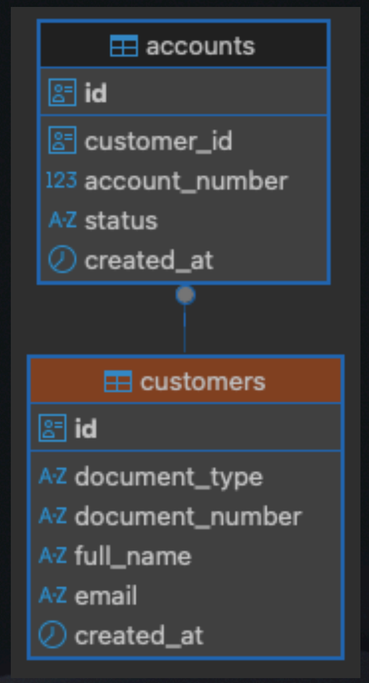
Configuración de contenedor PostgreSQL



docker-compose up -d

DataBase

SQL y Relaciones



Relaciones

```sql
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
);
```

```sql
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

```sql
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;
```

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

Trigger

Monorepo

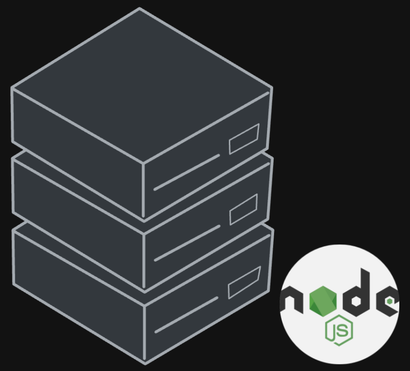
BackEnd

FrontEnd

Angular V20
Rxjs
TypeScript

Bootstrap
ng-bootstrap
ngx-toastr
ng-icons

ngx-translate



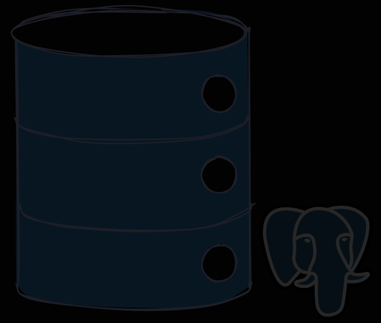
BackEnd

NodeJS
Express
Prisma

cors
pg
typescript

REST API
(get, post, patch)
i18n

Prisma
ORM



DataBase SQL

PostgreSQL

DBeaver

Function (account)
Trigger

Docker

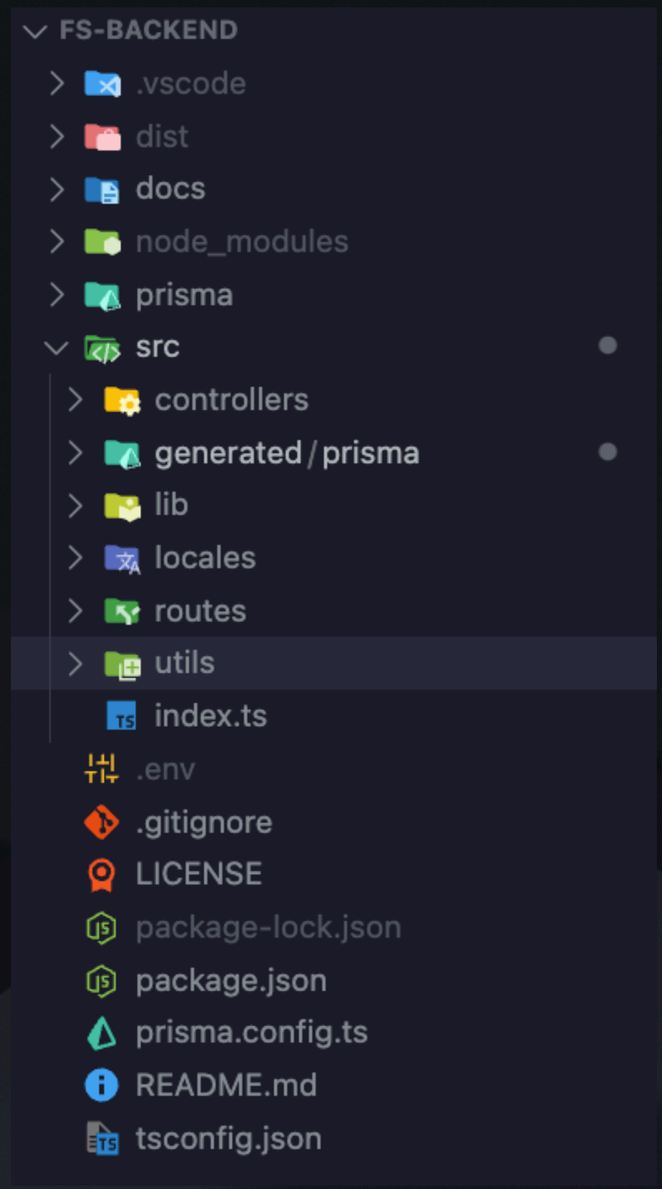
Railway
PAAS

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

Variable de connexion con DB

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

prisma.config.ts

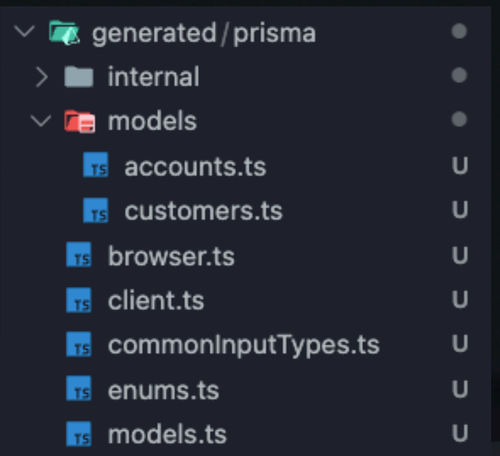
```ts
// This file was generated by Prisma, and assu
// npm install --save-dev prisma dotenv
import "dotenv/config";
import { defineConfig } from "prisma/config";

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

```
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:
}

/// 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[]
}
```

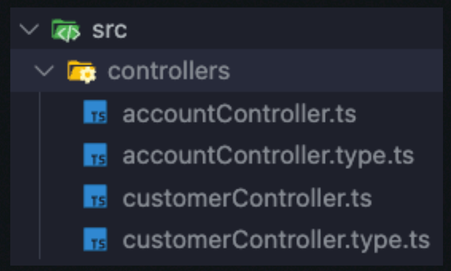
Datos de DB mapeados



Modelos y funciones generadas

BackEnd

Express: REST API's



Controladores

```ts
app.use(cors());
app.use(express.json());

// Routes
app.use('/api/customers', customerRoutes);
app.use('/api/accounts', accountRoutes);

const PORT = process.env.PORT ? Number(process.env.PORT) : 3000;
app.listen(PORT, () => {
  console.log(`🎸 Servidor corriendo en puerto ${PORT}`);
});
```

Endpoints para Cliente y Cuenta

```ts
const router: Router = Router();

router.get('/', getCustomers);
router.post('/', createCustomer);

export default router;
```

Métodos HTTP para Cliente

```ts
const router: Router = Router();

router.get('/', getAccounts);
router.post('/', createAccount);
router.patch('/:id/status', updateAccountStatus);

export default router;
```

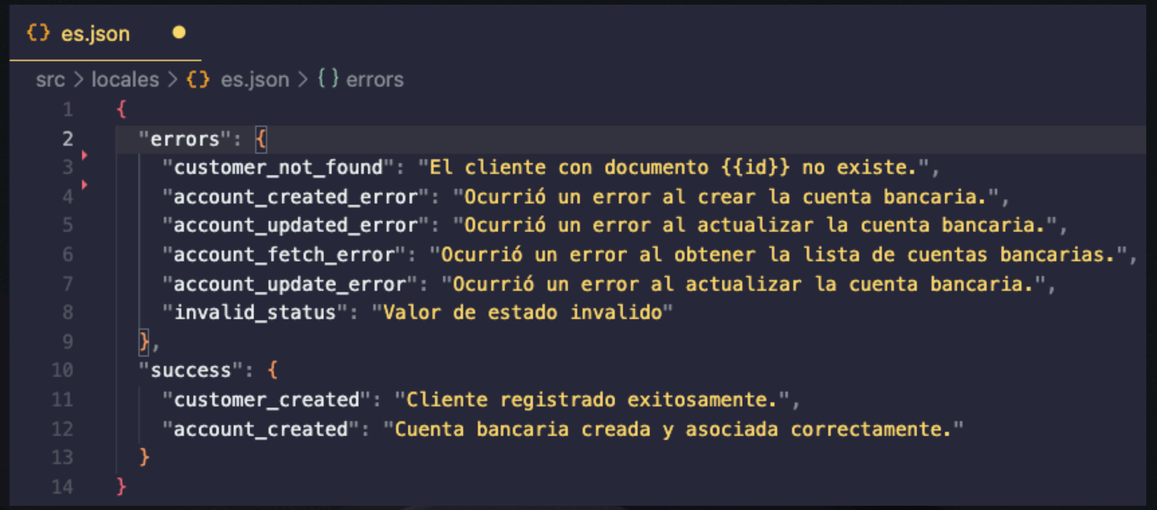
Métodos HTTP para Cuenta

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"),
    });
  }
```

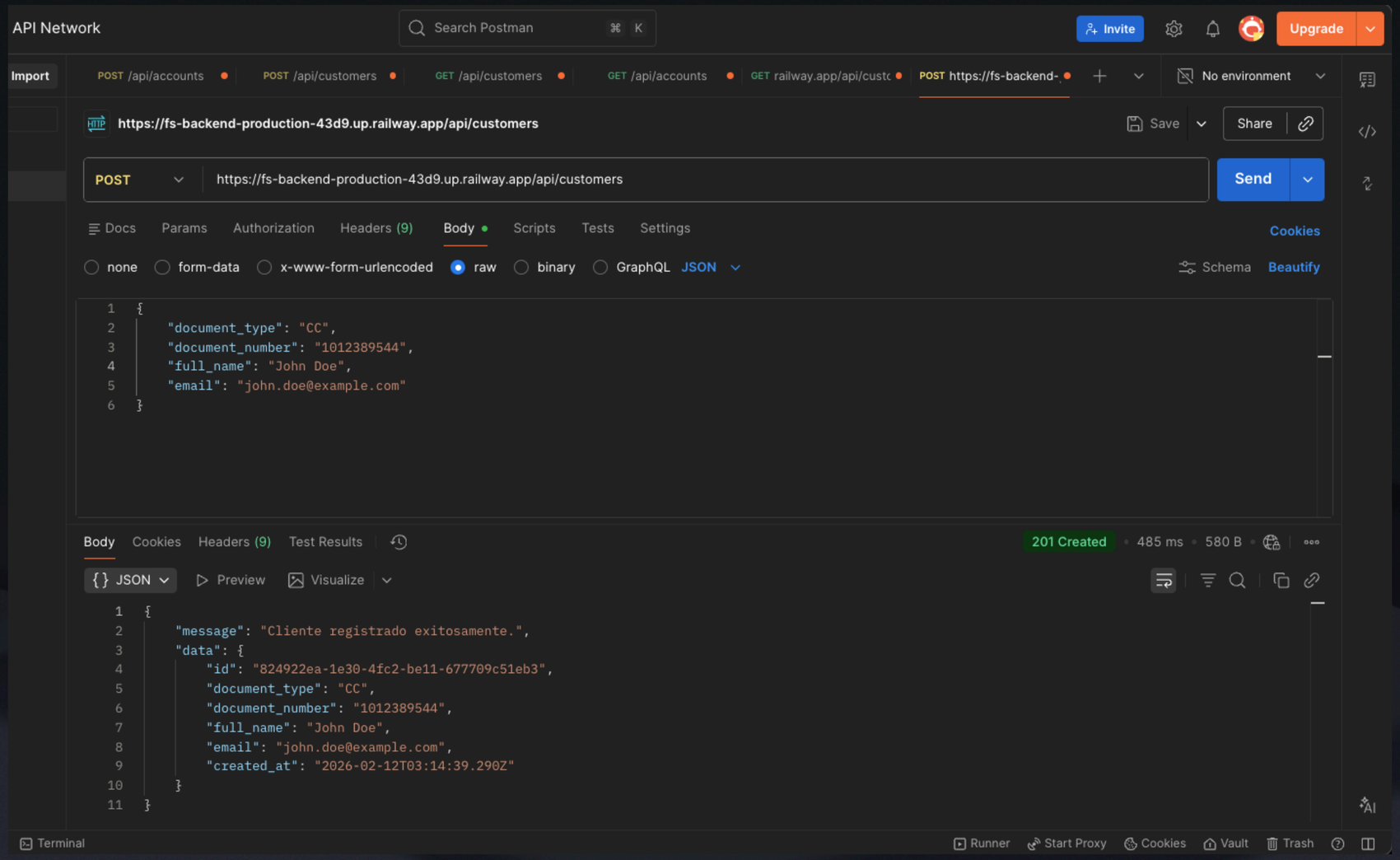
Utilidad de traducción



Textos centralizados por idioma

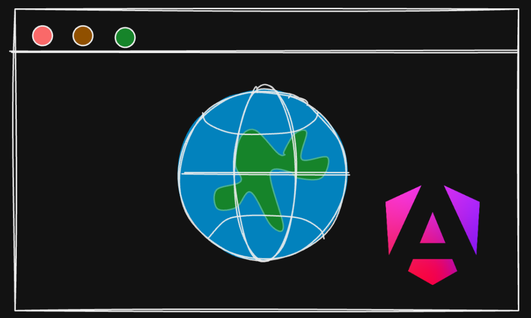
BackEnd

Peticiones y respuestas



Uso de Postman

Monorepo

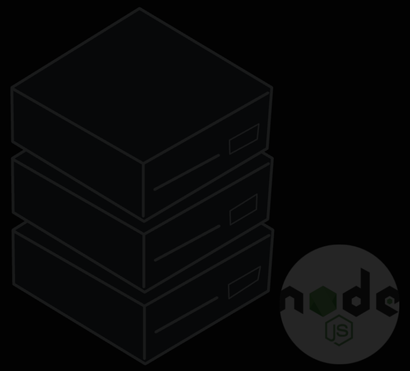


FrontEnd

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

Prism
ORM

FrontEnd

DataBase SQL

PostgreSQL

DBeaver

Function (account)
Trigger

Docker

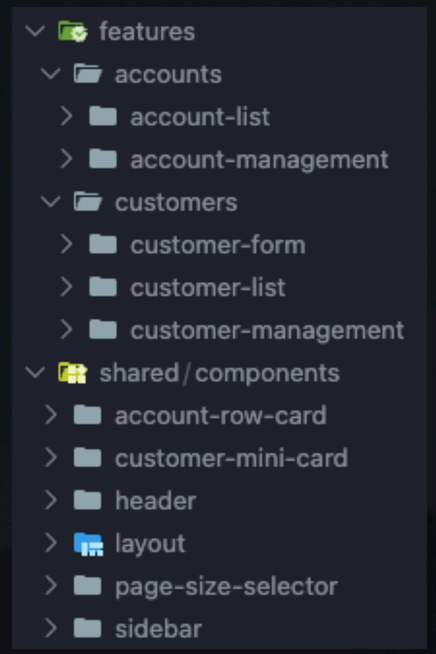
Railway
PAAS

FrontEnd

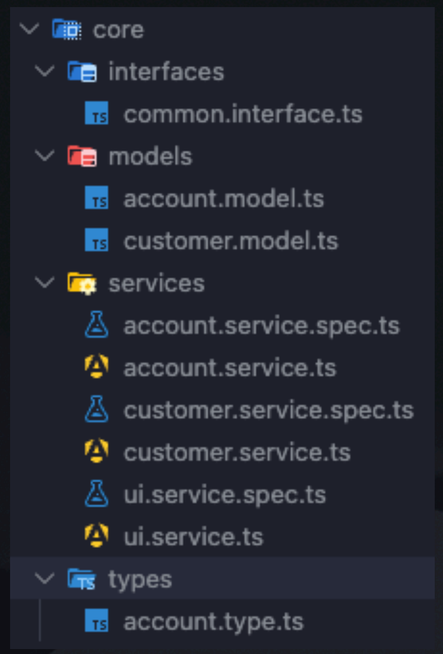
Configuración inicial

```
"dependencies": {
  "@angular/animations": "^20.3.16",
  "@angular/cdk": "^20.2.14",
  "@angular/common": "^20.3.0",
  "@angular/compiler": "^20.3.0",
  "@angular/core": "^20.3.0",
  "@angular/forms": "^20.3.0",
  "@angular/platform-browser": "^20.3.0",
  "@angular/router": "^20.3.0",
  "@ng-bootstrap/ng-bootstrap": "^19.0.1",
  "@ng-icons/bootstrap-icons": "^33.1.0",
  "@ng-icons/core": "^32.5.0",
  "@ngx-translate/core": "^17.0.0",
  "@ngx-translate/http-loader": "^17.0.0",
  "bootstrap": "^5.3.6",
  "ngx-toastr": "^19.1.0",
  "rxjs": "~7.8.0",
  "tslib": "^2.3.0"
},
"devDependencies": {
  "@angular/build": "^20.3.15",
  "@angular/cli": "^20.3.15",
  "@angular/compiler-cli": "^20.3.0",
  "@angular/localize": "^20.3.16",
  "@types/jasmine": "~5.1.0",
  "jasmine-core": "~5.9.0",
  "karma": "~6.4.0",
  "karma-chrome-launcher": "~3.2.0",
  "karma-coverage": "~2.2.0",
  "karma-jasmine": "~5.1.0",
  "karma-jasmine-html-reporter": "~2.1.0",
  "typescript": "~5.9.2"
}
```

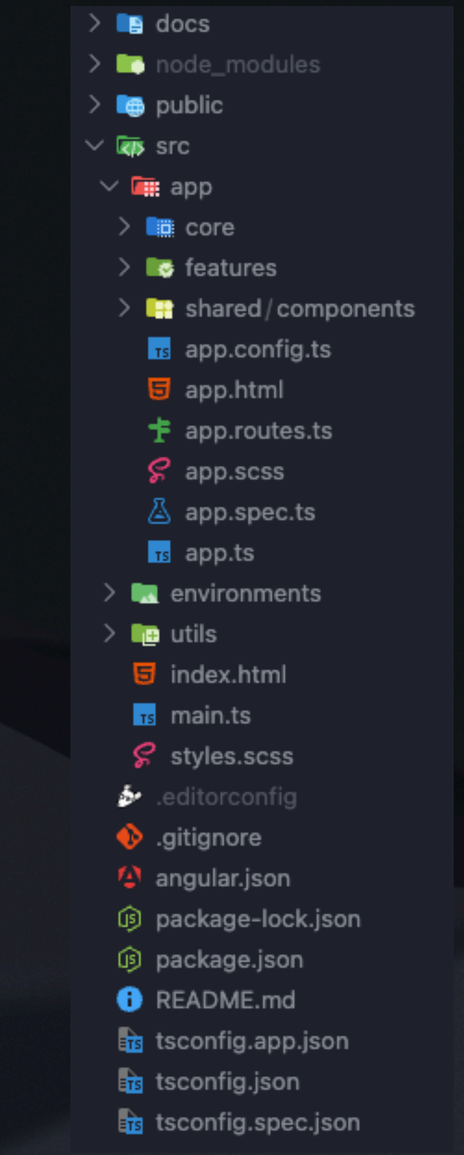
Dependencias y versiones



Componentes



Modelos, Tipos y Servicios



Estructura de archivos

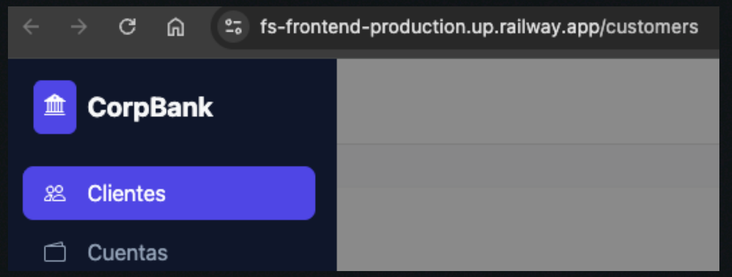
FrontEnd

Rutas

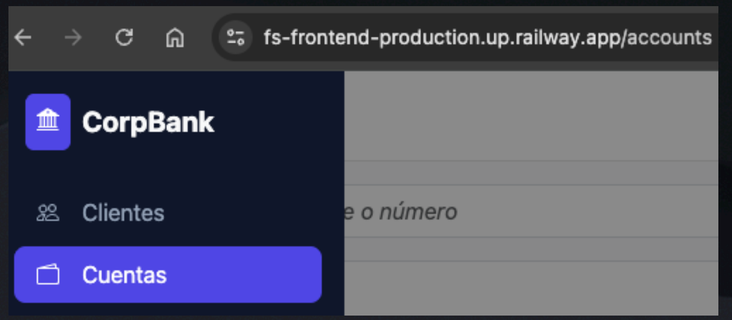
app.routes.ts M

src > app > app.routes.ts > ...

```ts
import { Routes } from '@angular/router';
import { Layout } from './shared/components/layout/layout';

export const routes: Routes = [
  {
    path: '',
    component: Layout,
    children: [
      { path: '', redirectTo: 'customers', pathMatch: 'full' },
      {
        path: 'customers',
        loadComponent: () =>
          import('./features/customers/customer-management/customer-management').then(
            (m) => m.CustomerManagement,
          ),
      },
      {
        path: 'accounts',
        loadComponent: () =>
          import('./features/accounts/account-management/account-management').then(
            (m) => m.AccountManagement,
          ),
      },
    ],
  },
  { path: '**', redirectTo: 'customers' },
];
```



Sección Clientes



Sección Cuentas

FrontEnd

Servicios

```ts
@Injectable({
  providedIn: 'root',
})
export class CustomerService {
  private readonly http = inject(HttpClient);
  private readonly apiUrl = `${environment.apiUrl}/customers`;

  private readonly customerCreatedSource = new Subject<void>();
  customerCreated$ = this.customerCreatedSource.asObservable();

  readonly #customers = signal<Customer[]>([]);
  customers = this.#customers.asReadonly();

  // GET customers
  getCustomers(…
  }

  // POST create customer
  createCustomer(customer: Partial<Customer>): Observable<any> {…
  }
}
```

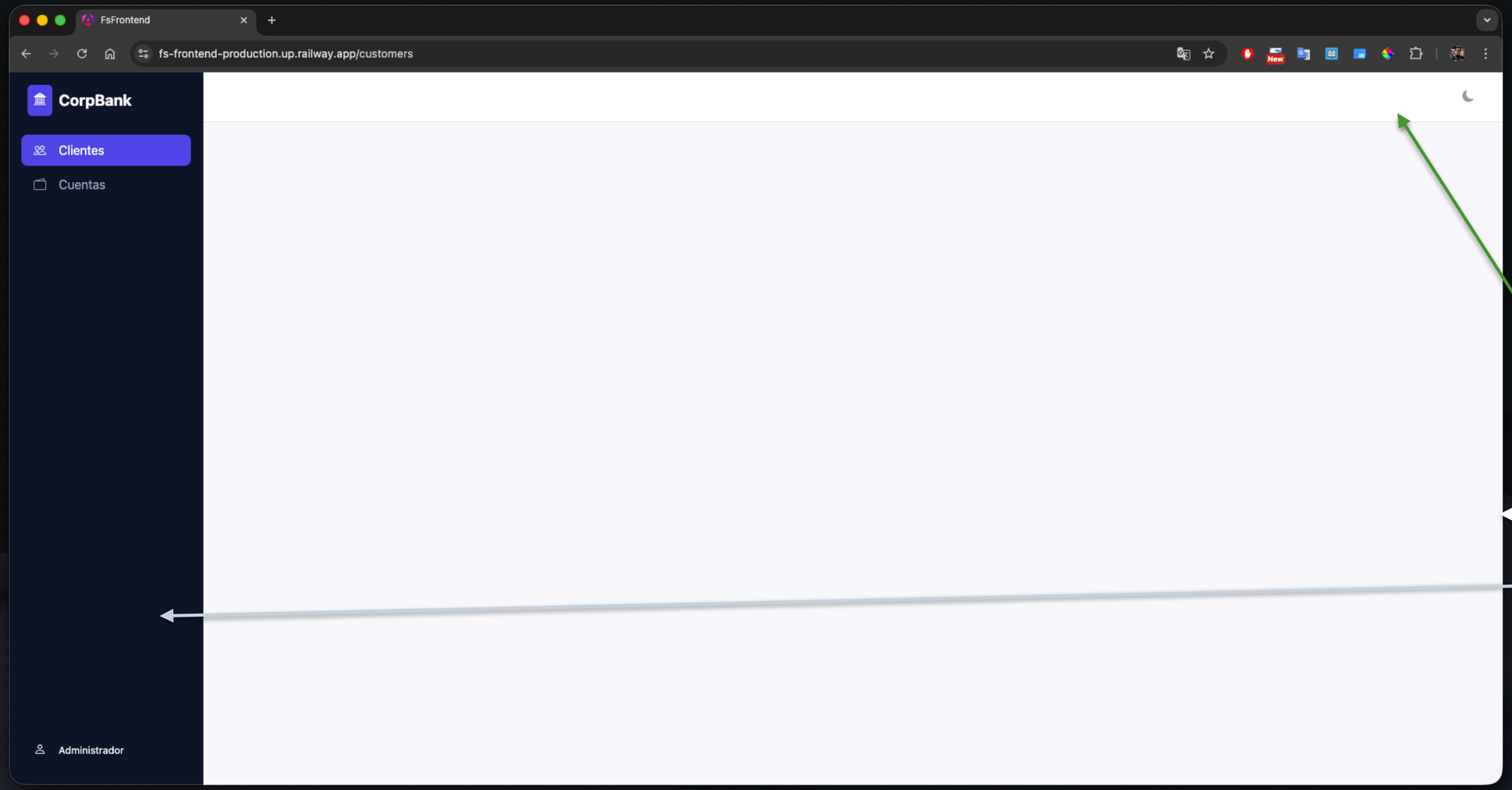
CustomerService

```ts
@Injectable({
  providedIn: 'root',
})
export class AccountService {
  private readonly http = inject(HttpClient);
  private readonly apiUrl = `${environment.apiUrl}/accounts`;

  private readonly accountChangedSource = new Subject<void>();
  accountChanged$ = this.accountChangedSource.asObservable();

  readonly #accounts = signal<Account[]>([]);
  accounts = this.#accounts.asReadonly();

  // GET accounts
  getAccounts(…
  }

  // POST create account
  createAccount(documentNumber: string = ''): Observable<any> {…
  }

  // PATCH status account
  updateStatus(accountId: string, newStatus: TAccountStatus): Observable<any> {…
  }
```

AccountService

FrontEnd

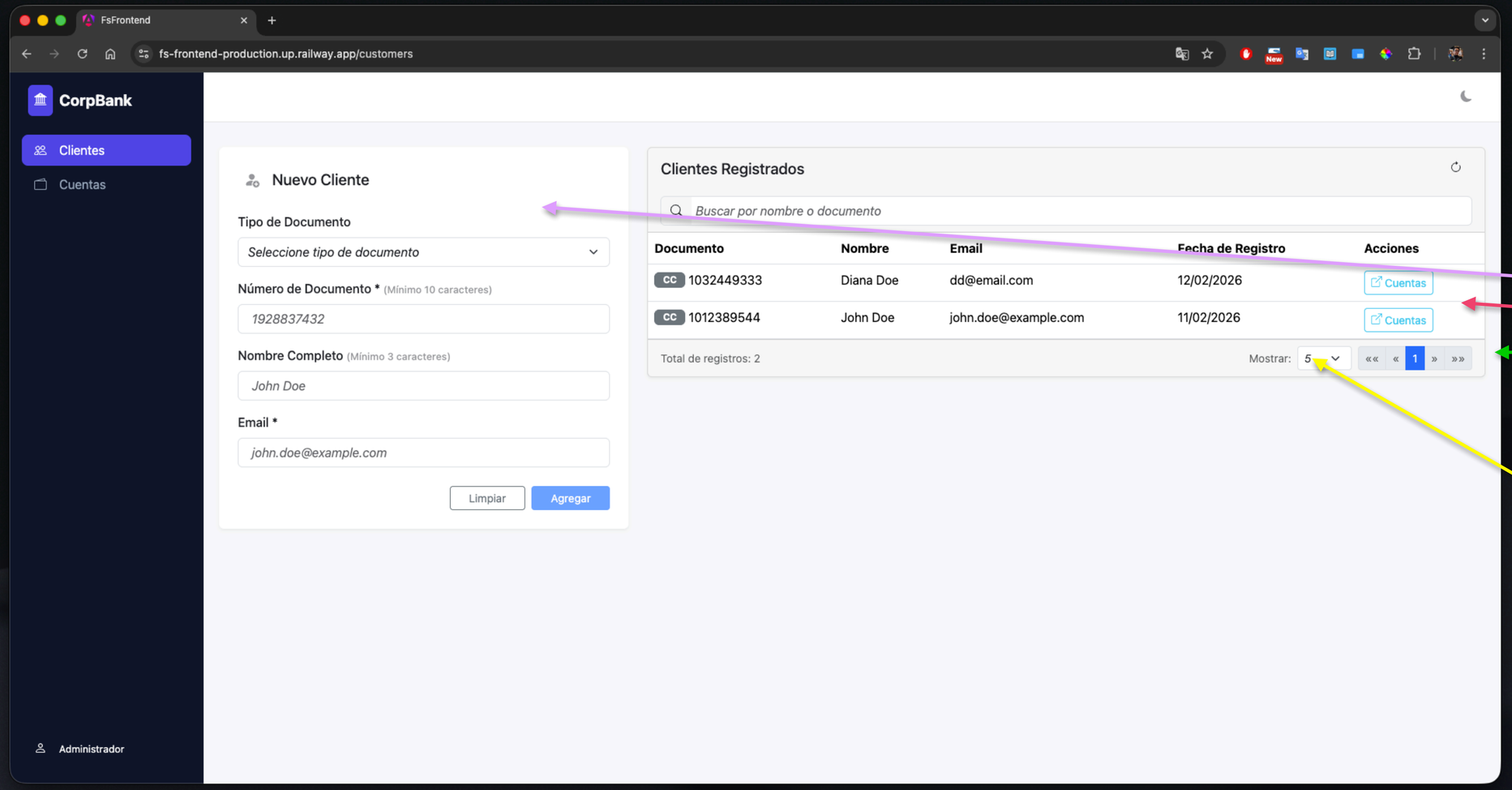
Componentes

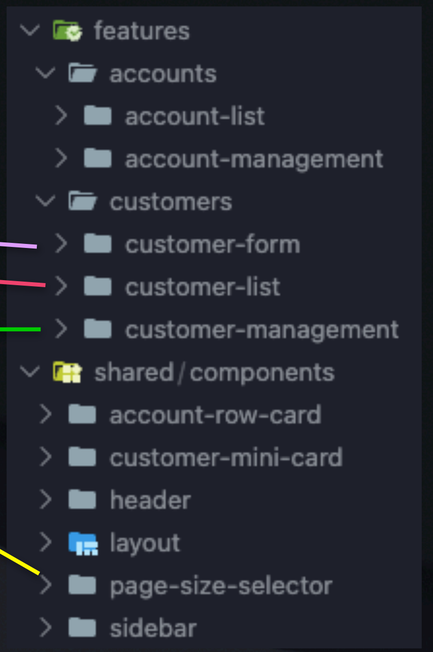


Layout principal

FrontEnd

Componentes

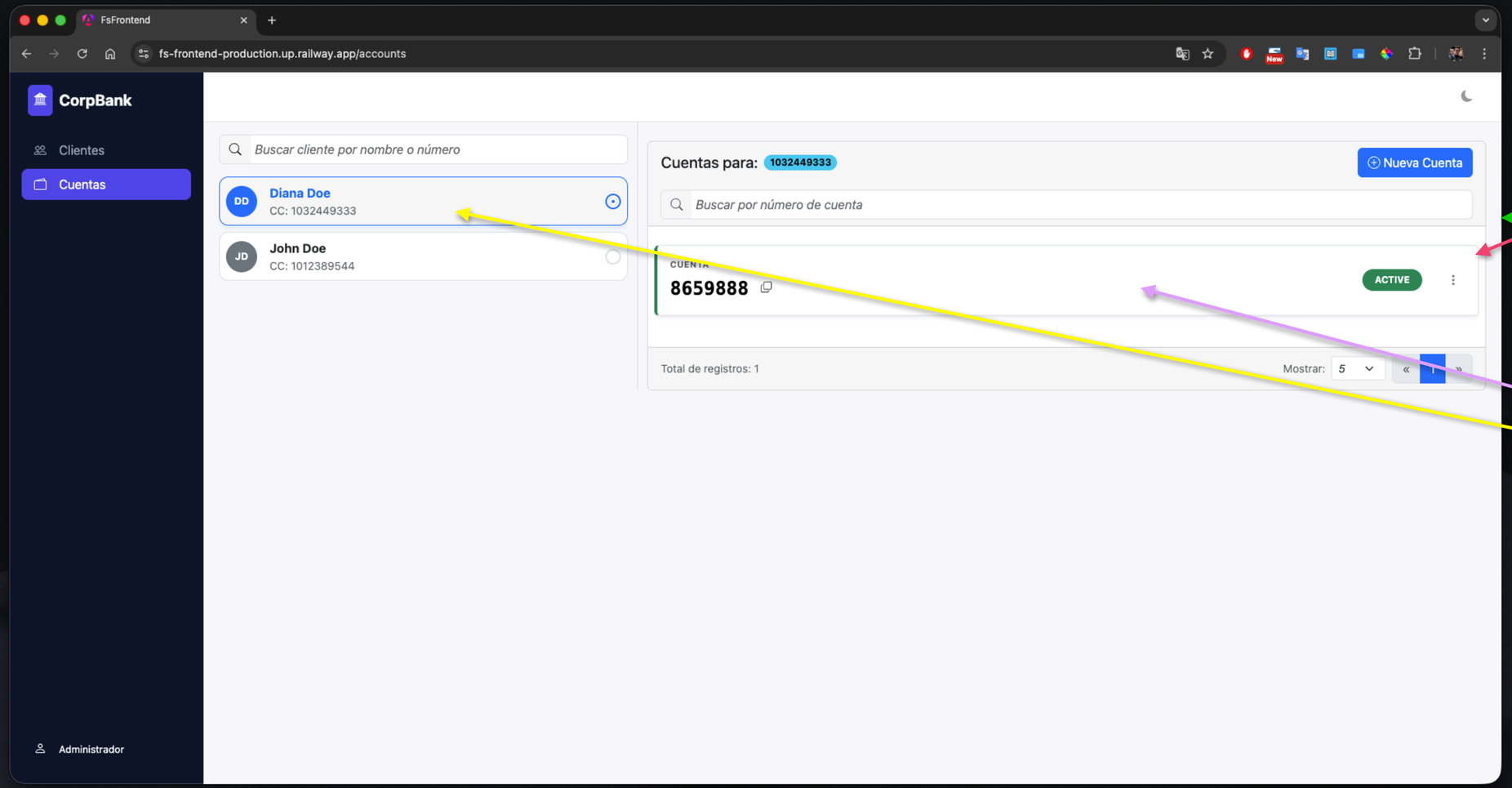


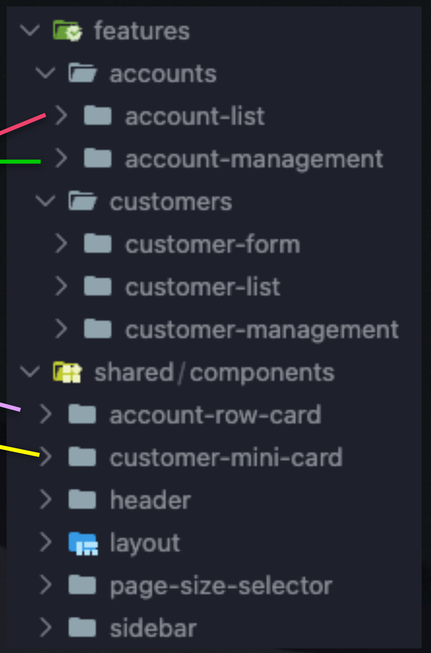


Sección Clientes

FrontEnd

Componentes

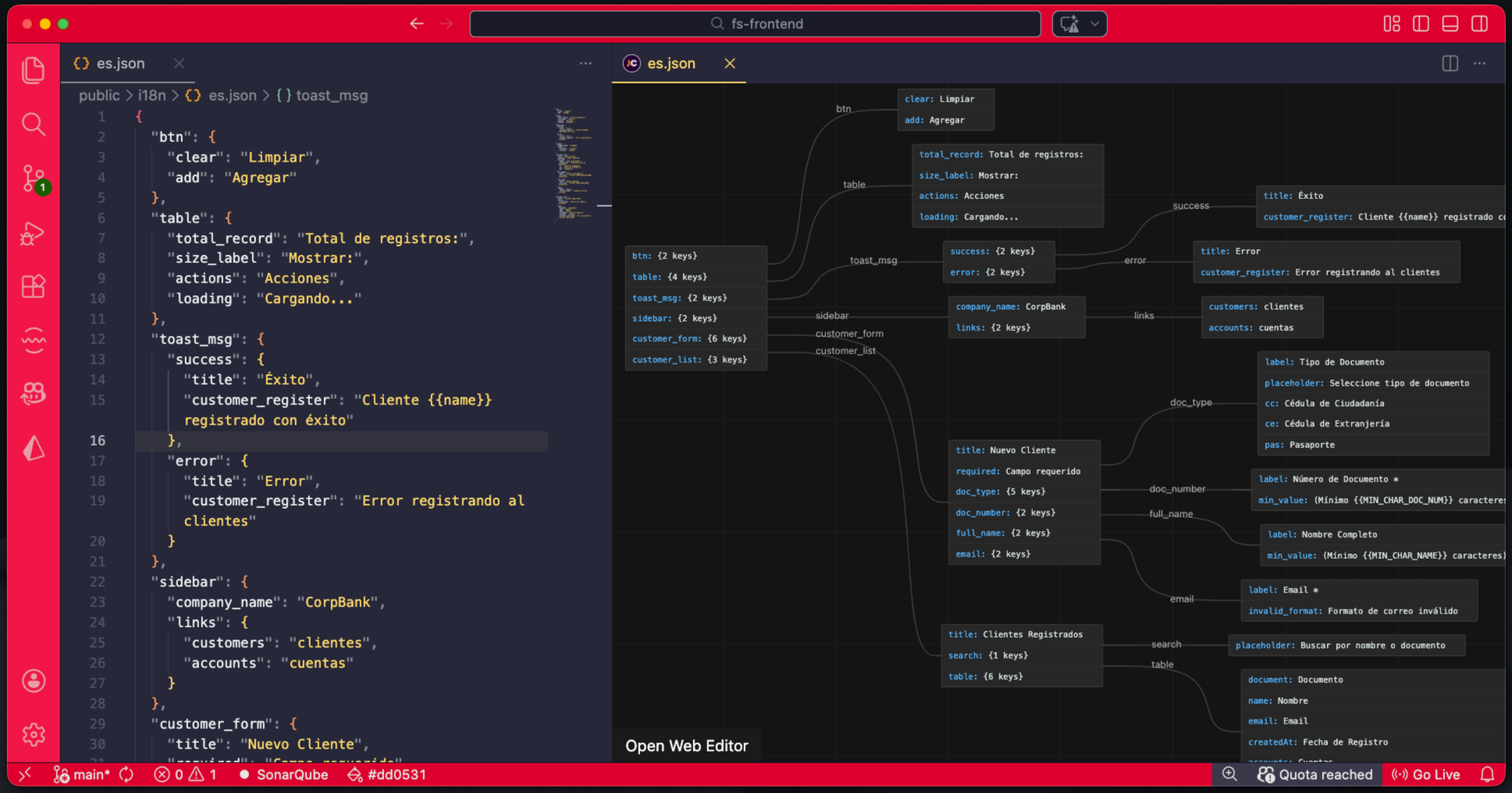




Sección Cuentas

FrontEnd

Internacionalización de textos



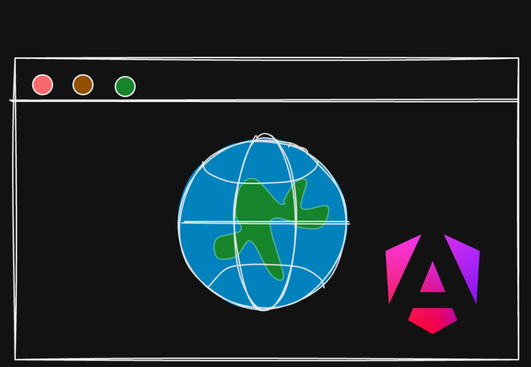

```
<div class="mb-4">
  <label class="form-label fw-medium" for="txt-email">
    {{ 'customer_form.email.label' | translate}}
  </label>
  <input
```

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

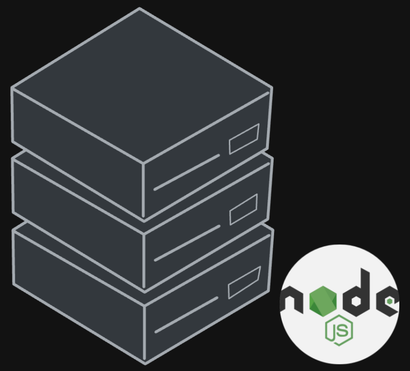


FrontEnd

Angular V20
Rxjs
TypeScript

Bootstrap
ng-bootstrap
ngx-toastr
ng-icons

ngx-translate



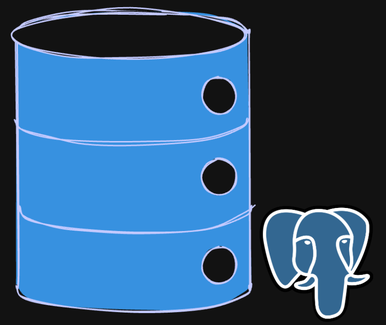
BackEnd

NodeJS
Express
Prisma

cors
pg
typescript

REST API
(get, post, patch)
i18n

Prisma
ORM



DataBase SQL

PostgreSQL

DBeaver

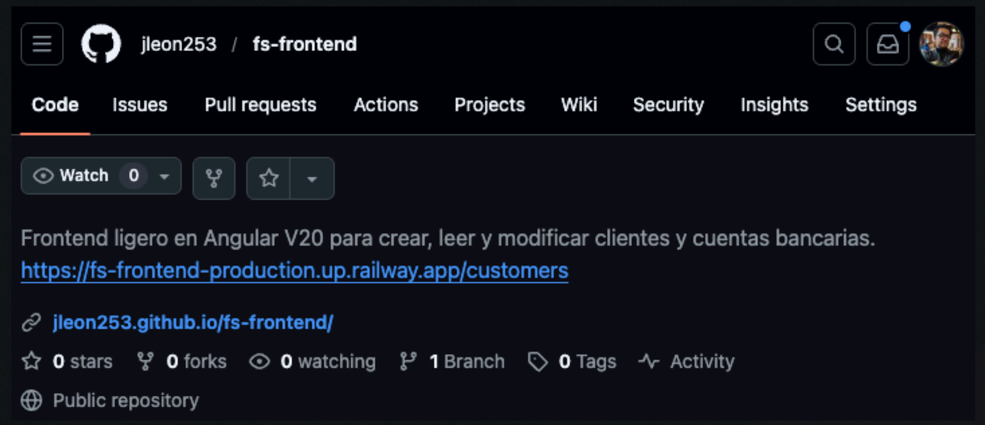
Function (account)
Trigger

Docker

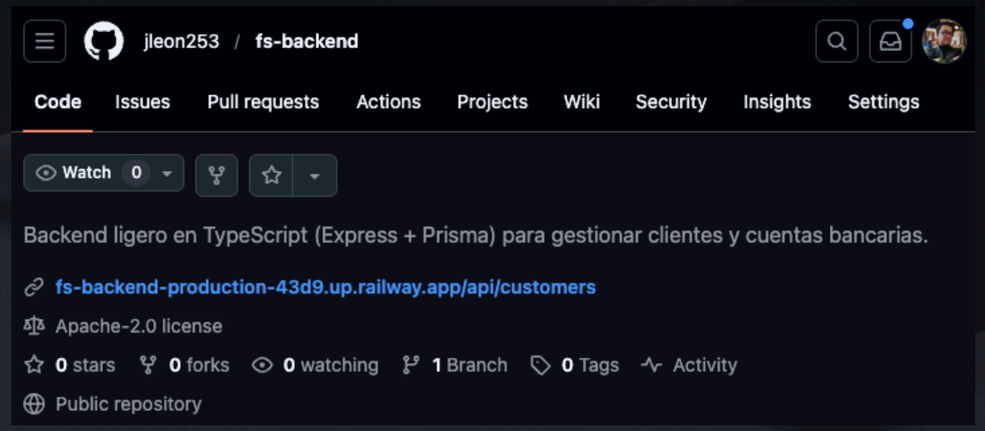
Railway
PAAS

GitHub y Railway

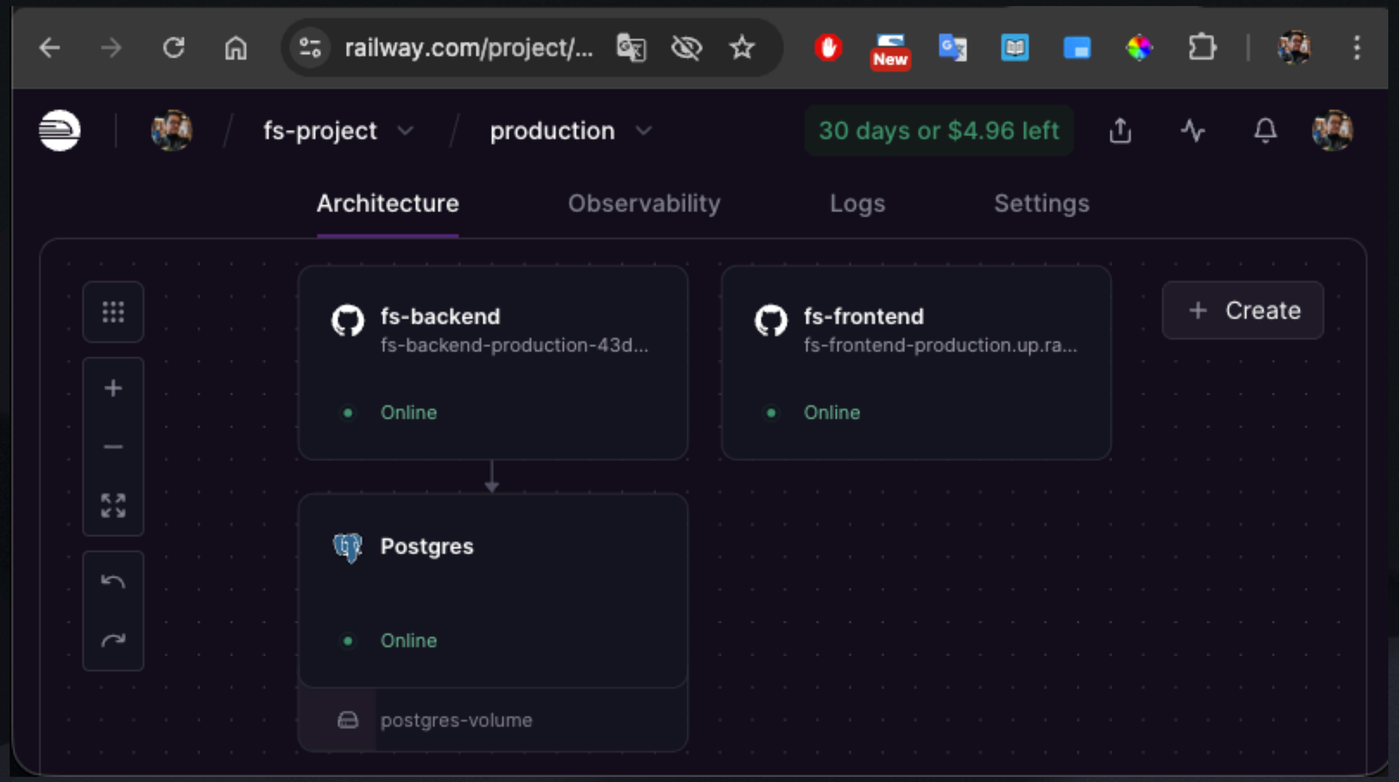
Control de versiones y Despliegue en la nube



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



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



Despliegue: DB, Back y Front

Respuestas sobre AWS

AWS

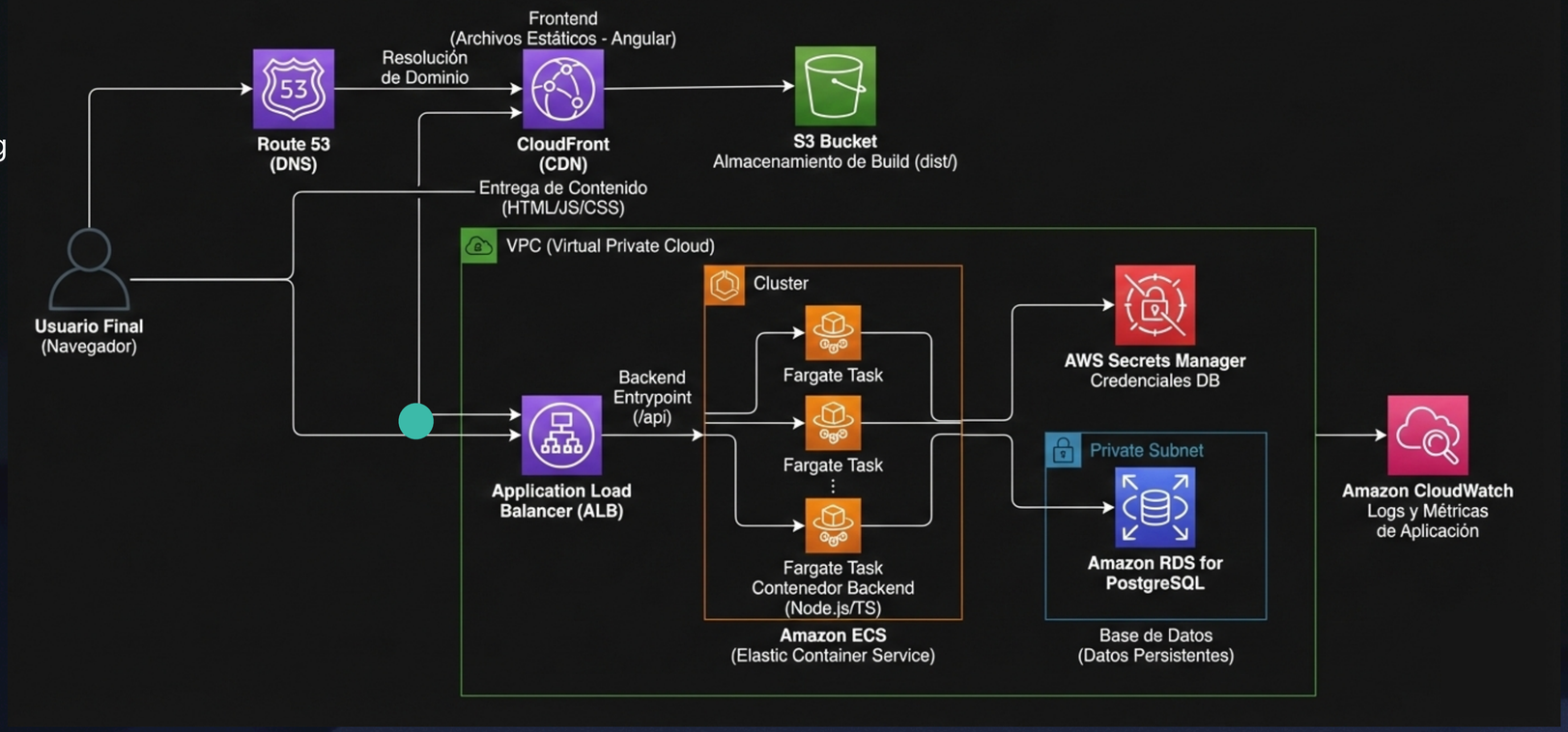
Respuestas

- 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
  - Analitica con Glue y Athena



Gracias