01 NEST SERVER REPORTS

Cascarón

- Creación del docker-compose.yml
- Tabla de empleados
- Conectar Prisma con Nest
- Creo el proyecto

> nest new reports-server

- Docker, Postgres y PgAdmin
- Creo en la raíz del proyecto el docker-compose.yml
- docker-compose.yml

```
services:
  db:
    container_name: postgres_database
    image: postgres:16.3
    volumes:
      - ./postgres:/var/lib/postgresql/data
    environment:
      - POSTGRES_PASSWORD=123456
    restart: always
    ports:
      - "5432:5432"

  pgAdmin:
    depends_on:
      - db
    image: dpage/pgadmin4:8.6
    volumes:
      - ./pgadmin:/var/lib/pgadmin
    ports:
      - "8080:80"
    environment:
      - PGADMIN_DEFAULT_PASSWORD=123456
      - PGADMIN_DEFAULT_EMAIL=superman@google.com
    restart: always
```

> docker compose up -d

- Voy a localhost:8080 puedo acceder a pgAdmin
  - Coloco mi usuario superman@google.com y el password 123456
- Configuro el servidor. Clico en servers/new Server
  - En la conexión tomo el nombre del contenedor postgres_database
  - Port 5432

    - Username postgres
    - Password 123456
- Ya tengo configurado pgAdmin con PostgreSQL
- Aquí tengo el archivo employees.sql para crear la db

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    position VARCHAR(50) NOT NULL,
    start_date DATE NOT NULL,
    work_time TIME NOT NULL,
    hours_per_day INT NOT NULL,
    work_schedule VARCHAR(50) NOT NULL
);

INSERT INTO employees (name, position, start_date, work_time, hours_per_day,
work_schedule)
VALUES
('Juan Pérez', 'Desarrollador', '2021-01-15', '09:00', 8, 'Lunes a Viernes, 9am -
5pm'),
('Ana Gómez', 'Diseñadora', '2020-03-22', '10:00', 6, 'Lunes a Viernes, 10am -
4pm'),
('Carlos Sánchez', 'Gerente', '2018-11-05', '08:00', 9, 'Lunes a Viernes, 8am -
5pm'),
('María López', 'Analista', '2019-07-11', '09:30', 7, 'Lunes a Viernes, 9:30am -
4:30pm'),
('Pedro Rodríguez', 'Programador', '2021-09-14', '11:00', 6, 'Lunes a Viernes,
11am - 5pm'),
('Lucía Fernández', 'Administrativa', '2020-12-01', '08:30', 8, 'Lunes a Viernes,
8:30am - 4:30pm'),
('José Martínez', 'Contador', '2017-05-19', '09:00', 8, 'Lunes a Viernes, 9am -
5pm'),
('Laura Ramírez', 'Desarrolladora', '2018-06-07', '10:00', 7, 'Lunes a Viernes,
10am - 5pm'),
('Miguel Torres', 'Soporte Técnico', '2021-03-16', '09:00', 8, 'Lunes a Viernes,
9am - 5pm'),
('Sara Morales', 'Recursos Humanos', '2019-09-23', '08:00', 7, 'Lunes a Viernes,
8am - 3pm'),
('David Vega', 'Desarrollador', '2022-02-14', '09:30', 7, 'Lunes a Viernes, 9:30am
- 4:30pm'),
('Elena Ortiz', 'Diseñadora', '2021-11-10', '10:30', 6, 'Lunes a Viernes, 10:30am
- 4:30pm'),
('Jorge Herrera', 'Gerente', '2016-04-18', '08:00', 9, 'Lunes a Viernes, 8am -
5pm'),
('Isabel Domínguez', 'Analista', '2019-02-05', '09:00', 8, 'Lunes a Viernes, 9am -
5pm'),
('Ricardo Ruiz', 'Programador', '2020-10-22', '10:00', 7, 'Lunes a Viernes, 10am -
5pm'),
('Patricia Flores', 'Administrativa', '2018-08-30', '08:30', 8, 'Lunes a Viernes,
8:30am - 4:30pm'),
('Roberto Castillo', 'Contador', '2017-12-12', '09:00', 8, 'Lunes a Viernes, 9am -
5pm'),
```

```
('Adriana Reyes', 'Desarrolladora', '2021-06-25', '09:30', 7, 'Lunes a Viernes,
9:30am - 4:30pm'),
('Santiago García', 'Soporte Técnico', '2020-01-13', '08:00', 8, 'Lunes a Viernes,
8am - 4pm'),
('Verónica Ríos', 'Recursos Humanos', '2019-04-17', '09:00', 7, 'Lunes a Viernes,
9am - 4pm');
```

- Creo la carpeta de queries fuera de src para ir almacenándolos
- En pgAdmin, clico el botón de Quey Tool
- Debo estar en postgres/postgres@Reports database
- Pego el código y le doy alk play para ejecutar
- Si en Tables clico refresh veré la tabla de empleados
- Si en el query coloco

```
select * from employees
```

- Obtengo toda la data

Conectar Nest con prisma

- Instalo prisma

```
npm i prisma
```

- o puedo usar el CLI

```
npx prisma init
```

- Configuro la .env para la DB

```
DATABASE_URL="postgresql://postgres:123456@localhost:5432/postgres"
```

- Instalo el cliente

```
npm i @prisma/client
```

- Genero un proyecto dentro de nest-report-server sin los endpoints del CRUD

```
nest g res basic-reports --no-spec
```

- Hago la conexión en el service

```
import { Injectable, OnModuleInit } from '@nestjs/common';
import { PrismaClient } from '@prisma/client';

@Injectable()
```

```
export class BasicReportsService extends PrismaClient implements OnModuleInit {
  async onModuleInit() {
    await this.$connect();
    // console.log('Connected to the database');
  }

  async hello() {
    return this.employees.findFirst();
  }
}
```

- Ejecuto **npx prisma db pull**
- Esto genera un schema basado en lo que hay en mi db a través del string de conexión
- En el schema obtengo esto

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model employees {
  id            Int      @id @default(autoincrement())
  name          String   @db.VarChar(100)
  position      String   @db.VarChar(50)
  start_date    DateTime @db.Date
  work_time     DateTime @db.Time(6)
  hours_per_day Int
  work_schedule String   @db.VarChar(50)
}
```

- Genero el cliente con **npx prisma generate**
- Levanto el server

```
npm run start:dev
```

- Creamos el README

```
# Ejecutar en Dev

1. Clonar el repositorio
2. Instalar dependencias `npm install`
3. Clonar `env.template` y renombrar a `.env` y completar las variables de entorno
en .env
4. Levantar la base de datos `docker compose up -d`
```

```
5. Generar el Prisma client `npx prisma generate`
6. Ejecutar proyecto `npm run start:dev`
```

---

02 NEST REPORTS CONSTANCIA DE EMPLEADOS

- Vamos a crear un reporte con el logo de la compañía, donde empleado tal certifique que trabaja de esto en la compañía con su footer, header..
- Nos haremos una idea de como funciona pdfmake, básicamente es línea por línea
- Cuando hay dos elementos en una linea, entonces separamos por columnas
- Veremos:
  - Estructura del contenido del reporte
  - Crear encabezados y pies de página
  - Trabajar con imágenes desde el backend
  - Formatear fechas
  - Columnas
  - Cragar data en el reporte
  - Estilos personalizados
  - Secciones del reporte de forma reutilizable
  - Crear una constancia laboral

Continuación

- Con docker compose up -d levanto las imagenes de postgres y pgAdmin
- Si no tuviera la db (sin las carpetas de postgres y pgAdmin) las creará de nuevo
- Ejecuto npx primsa generate
- Ahora ya puedo trabajar con prisma
- Ejecuto npm run start:dev
- Si hago una llamada a un endpoint y tengo un error 500 porque borré la tabla, la carpeta de postgres, ingreso de nuevo en el 8080 y entro en pgAdmin
- Registro un nuevo servidor, en conexiones, el nombre del host es el nombre del container de Docker (postgres_database)
- Puerto 5432, usuario postgres, password 123456
- En Database/ databases/ schemas/ tables
- Vuelvo a construir la tabla con el query tool, pegando el código entregado actualmente del archivo employees.sql

---

PdfMake

- Para consultar la documentación, hay varios ejemplos en pdfmake, documentation, Server side, examples

> npm i pdfmake npm i -D @types/pdfmake

- Creo la carpeta fonts en la raíz del proyecto y copio todas las fuentes del gist

- Creo un módulo printer dentro de src con su servicio con **nest g mo printer** y **nest g service printer**
- En el módulo de printer exporto el servicio

```
import { Module } from '@nestjs/common';
import { PrinterService } from './printer.service';

@Module({
  providers: [PrinterService],
  exports: [PrinterService],
})
export class PrinterModule {}
```

- Me aseguro de que el módulo de Printer esté importado en app.module

```
import { Module } from '@nestjs/common';
import { BasicReportsModule } from './basic-reports/basic-reports.module';
import { PrinterModule } from './printer/printer.module';

@Module({
  imports: [BasicReportsModule, PrinterModule],
})
export class AppModule {}
```

- Hago la inyección de las fuentes en la instancia de la impresora en el printer.service
- Uso las interfaces de pdfmake. BufferOptions me servirá para configuraciones posteriores
- PDFKit debería estar de manera global con los @types/pdfmake

```
import { Injectable } from '@nestjs/common';
import PdfPrinter from 'pdfmake';
import { BufferOptions, TDocumentDefinitions } from 'pdfmake/interfaces';

const fonts = {
  Roboto: {
    normal: 'fonts/Roboto-Regular.ttf',
    bold: 'fonts/Roboto-Medium.ttf',
    italics: 'fonts/Roboto-Italic.ttf',
    bolditalics: 'fonts/Roboto-MediumItalic.ttf',
  },
};

@Injectable()
export class PrinterService {
  private printer = new PdfPrinter(fonts); //defino la impresora

  createPdf(
    docDefinition: TDocumentDefinitions,//interfaz de pdfmake
    options: BufferOptions = {}, //interfaz de pdfmake
  ): PDFKit.PDFDocument {
```

```
    return this.printer.createPdfKitDocument(docDefinition, options); //uso la
impresora pra crear el pdf
  }
}
```

- En el basic-reports.controller inyecto el printerService
- Llamo al super porque prisma me lo exije
- En el caso de uso getHelloWorldreport le paso el name y lo guardo en docDefinition
- Uso el método createPdf del servicio de PrinterService y le paso el docdefinition con el name
    - Lo guardo en doc
- Retorno el doc

```
import { Injectable, NotFoundException, OnModuleInit } from '@nestjs/common';
import { PrismaClient } from '@prisma/client';
import { PrinterService } from 'src/printer/printer.service';
import {
  getEmploymentLetterByIdReport,
  getEmploymentLetterReport,
  getHelloWorldReport,
} from 'src/reports';

@Injectable()
export class BasicReportsService extends PrismaClient implements OnModuleInit {
  async onModuleInit() {
    await this.$connect();
    // console.log('Connected to the database');
  }
  constructor(private readonly printerService: PrinterService) {
    super();
  }

  hello() {               //en la carpeta reports está getHelloWorldReport
    const docDefinition = getHelloWorldReport({
      name: 'Fernando Herrera',
    });

    const doc = this.printerService.createPdf(docDefinition); //creo el pdf

    return doc; //retorno el pdf
  }
}
```

- **getHelloWorldReport** está en la carpeta src/reports
- Le paso el name en las options (creo una interfaz), devolverá algo de tipo TDocumentdefinitions
- Extraigo el name con desestructuración
- En el objeto introduzco un template stray dentro del array de content y le paso el name
- Retorno el doc
- reports/hello-world.ts

```
import type { TDocumentDefinitions } from 'pdfmake/interfaces';

interface ReportOptions {
  name: string;
}

export const getHelloWorldReport = (
  options: ReportOptions,
): TDocumentDefinitions => {
  const { name } = options;

  const docDefinition: TDocumentDefinitions = {
    content: [`Hola ${name}`],
  };

  return docDefinition;
};
```

- Y desde el basic-reports.controller llamo al servicio
- Utilizo **@Res** para obtener la Response y poder setear el content-type y demás
- Guardo el doc que obtengo con el basicReportsService.hello que llama al caso de uso getHelloWorldReport y genera el pdf con el PrinterService
- Con la response seteo el contentType
- Uso el pdf que he obtenido y con .info.Title le pongo un título
- Uso un pipe para enlazar la respuesta con el pdf
- Cierro el pdfDoc

```
import { Controller, Get, Param, Res } from '@nestjs/common';
import { BasicReportsService } from './basic-reports.service';
import { Response } from 'express';

@Controller('basic-reports')
export class BasicReportsController {
  constructor(private readonly basicReportsService: BasicReportsService) {}

  @Get()
  async hello(@Res() response: Response) {
    const pdfDoc = this.basicReportsService.hello();

    response.setHeader('Content-Type', 'application/pdf'); //le digo que el
content-type es un pdf
    pdfDoc.info.Title = 'Hola-Mundo.pdf'; //le pongo un título, el .pdf no es
necesario
    pdfDoc.pipe(response); //vamos a enlazar la respuesta con el pdf
    pdfDoc.end(); //cerramos
  }
}
```

- Para que el pdfmake_1.default is not a constructor no dé problemas, debo colocar en el **tsconfig el esModuleInterop: true**

Cómo funciona pdfmake

- Cuando observamos un documento hecho con pdfmake (o cualquiera en realidad), podemos analizarlo como un html y dividirlo en lineas horizontales (las verticales vendrán después)
- Tengo el header, el h1 centrado que puede estar dentro de un section, los párrafos que pueden estar en una sola linea y separarlos con /n, un section o un div con el atentamente(1 linea), debajo el nombre del empleador (otra linea) como un arreglo de string con /n para crear una linea debajo de la otra, y un footer
- Podemos crear un sistema de columnas

Constancia de empleo - Reporte

- Creo la carpeta assets donde guardo las imágenes que usaré para el reporte y tengo el texto que quiero imprimir en el documento
- Empecemos por el controlador, creando employmentLetter
- Es básicamente lo mismo que el ejemplo anterior

```
@Get('employment-letter')
 async employmentLetter(@Res() response: Response) {
   const pdfDoc = this.basicReportsService.employmentLetter();

   response.setHeader('Content-Type', 'application/pdf');
   pdfDoc.info.Title = 'Employment-Letter';
   pdfDoc.pipe(response);
   pdfDoc.end();
 }
```

- En basicreportsService guardo el caso de uso y uso printerService para generar el pdf

```
employmentLetter() {
  const docDefinition = getEmploymentLetterReport();
  const doc = this.printerService.createPdf(docDefinition);
  return doc;
}
```

- El caso de uso en src/reports/getemploymentLetterReport
- Creo unos estilos para el header de tipo StyleDictionary de pdfmake
- Al docDefinition de tipo TDocumentDefinitions le paso los estilos en la propiedad styles
    - Le paso los márgenes de la página
    - En el headerSection pongo el showLogo y showdate en true
    - Añado el content que es un arreglo de objetos con las diferentes secciones

    - Le paso el estilo de cada párrafo con la propiedad style
- Retorno el documento

```
import type { StyleDictionary, TDocumentDefinitions } from 'pdfmake/interfaces';
import { headerSection } from './sections/header.section';

const styles: StyleDictionary = {
  header: {
    fontSize: 22,
    bold: true,
    alignment: 'center', //alignment es left por defecto
    margin: [0, 60, 0, 20], //left, top, right, bottom. Añado los margins
  },
  body: {
    alignment: 'justify',
    margin: [0, 0, 0, 70],
  },
  signature: {
    fontSize: 14,
    bold: true,
    // alignment: 'left',
  },
  footer: {
    fontSize: 10,
    italics: true,
    alignment: 'center',
    margin: [0, 0, 0, 20],
  },
};

export const getEmploymentLetterReport = (): TDocumentDefinitions => {
  const docDefinition: TDocumentDefinitions = {
    styles: styles,
    pageMargins: [40, 60, 40, 60], //creo los márgenes del documento

  //para mostrar el logo y la data
    header: headerSection({
      showLogo: true,
      showDate: true,
    }),

    content: [
      {
        text: 'CONSTANCIA DE EMPLEO',
        style: 'header',
      },
      {
        text: `Yo, [Nombre del Empleador], en mi calidad de [Cargo del Empleador]
de [Nombre de la Empresa], por medio de la presente certifico que [Nombre del
Empleado] ha sido empleado en nuestra empresa desde el [Fecha de Inicio del
Empleado]. \n\n
        Durante su empleo, el Sr./Sra. [Nombre del Empleado] ha desempeñado el
```

```
cargo de [Cargo del Empleado], demostrando responsabilidad, compromiso y
habilidades profesionales en sus labores.\n\n
        La jornada laboral del Sr./ Sra. [Nombre del Empleado] es de [Número de
Horas] horas semanales, con un horario de [Horario de Trabajo], cumpliendo con las
políticas y procedimientos establecidos por la empresa.\n\n
        Esta constancia se expide a solicitud del interesado para los fines que
considere conveniente. \n\n`,
        style: 'body',
      },
      { text: `Atentamente`, style: 'signature' },
      { text: `[Nombre del Empleador] `, style: 'signature' },
      { text: `[Cargo del Empleador]`, style: 'signature' },
      { text: `[Nombre de la Empresa]`, style: 'signature' },
      { text: `[Fecha de Emisión]`, style: 'signature' },
    ],

    footer: {
      text: 'Este documento es una constancia de empleo y no representa un
compromiso laboral.',
      style: 'footer',
    },
  };

  return docDefinition;
};
```

- En reports/sections/ tengo el header.section para mostarr el logo y la data
- Creo el logo de tipo Content de pdfmake
- En la interfaz HeaderOptions añado las propiedades, todas opcionales
- Le paso las options al método que devolverá algo de tipo Content
- Desestructuro de las options lo que sea que haya y seteo por defecto a true el showLogo y showDate
- Con un ternario evalúo si viene el showLogo en true le paso el logo, si no null
- Guardo en headerDate el showDate también usando un ternario
    - Si viene uso el DateFormatter que he creado para formatear la fecha y añadirlo a la izquierda
- Hago algo parecido guardando el title en headerTitle, pasándole en el texto el title y el estilo
- Regreso en un objeto arreglo de la propiedad columns con las constantes que he creado
- Estas columns crea las columnas en el documento, en este caso 3
- header.section

```
import { Content } from 'pdfmake/interfaces';
import { DateFormatter } from 'src/helpers';

const logo: Content = {
  image: 'src/assets/tucan-code-logo.png',
  width: 100, //le doy un tamaño
  height: 100,
  alignment: 'center', //la coloco en el centro de lo que será la primera columna
  margin: [0, 0, 0, 20],
};
```

```
interface HeaderOptions {
  title?: string;
  subTitle?: string;
  showLogo?: boolean;
  showDate?: boolean;
}

export const headerSection = (options: HeaderOptions): Content => {
  const { title, subTitle, showLogo = true, showDate = true } = options;

  const headerLogo: Content = showLogo ? logo : null;
  const headerDate: Content = showDate
    ? {
        text: DateFormatter.getDDMMMMYYYY(new Date()),
        alignment: 'right',
        margin: [20, 20],
      }
    : null;

  const headerTitle: Content = title
    ? {
        text: title,
        style: {
          bold: true,
        },
      }
    : null;

  return {
    columns: [headerLogo, headerTitle, headerDate],
  };
};
```

- El dateFormatter lo tengo en src/helpers

```
export class DateFormatter {
  static formatter = new Intl.DateTimeFormat('es-ES', {
    year: 'numeric',
    month: 'long',
    day: '2-digit',
  });

  static getDDMMMMYYYY(date: Date): string {
    return this.formatter.format(date);
  }
}
```

- Documentación oficial
- https://pdfmake.github.io/docs/0.1/document-definition-object/headers-footers/

Cargar información del empleado

- En el basics-reports.controller

```
@Get('employment-letter/:employeeId')
async employmentLetterById(
  @Res() response: Response,
  @Param('employeeId') employeeId: string,
) {
  const pdfDoc =
    await this.basicReportsService.employmentLetterById(+employeeId);

  response.setHeader('Content-Type', 'application/pdf');
  pdfDoc.info.Title = 'Employment-Letter';
  pdfDoc.pipe(response);
  pdfDoc.end();
}
```

- En el basic-reports.service compruebo que exista el employee
- Genero el empleado con el caso de uso de getEmployementLetterByIdReport

```
async employmentLetterById(employeeId: number) {
  const employee = await this.employees.findUnique({
    where: {
      id: employeeId,
    },
  });

  if (!employee) {
    throw new NotFoundException(`Employee with id ${employeeId} not found`);
  }

  const docDefinition = getEmploymentLetterByIdReport({
    employerName: 'Fernando Herrera',
    employerPosition: 'Gerente de RRHH',
    employeeName: employee.name,
    employeePosition: employee.position,
    employeeStartDate: employee.start_date,
    employeeHours: employee.hours_per_day,
    employeeWorkSchedule: employee.work_schedule,
    employerCompany: 'Tucan Code Corp.',
  });

  const doc = this.printerService.createPdf(docDefinition);

  return doc;
}
```

- En reports/emplyment-letter-bi-id.reports creo la interfaz de los parámetros que debo pasarle

- Creo los estilos
- Creo el método, le paso el ReportValues, devuelve algo de tipo TDocumentDefinitions
- Desestructuro las propiedades
- Creo el docDefinition que es de tipo TDocumentDefinitions y le paso los estilos
- Utilizo la misma plantilla que cree antes y usando el template literal le paso las variables
- le añado los estilos correspondientes a cada objeto que he creado previamente en el StyleDictionary

```
import type { StyleDictionary, TDocumentDefinitions } from 'pdfmake/interfaces';
import { headerSection } from './sections/header.section';
import { DateFormatter } from 'src/helpers';

interface ReportValues {
  employerName: string;
  employerPosition: string;
  employeeName: string;
  employeePosition: string;
  employeeStartDate: Date;
  employeeHours: number;
  employeeWorkSchedule: string;
  employerCompany: string;
}

const styles: StyleDictionary = {
  header: {
    fontSize: 22,
    bold: true,
    alignment: 'center',
    margin: [0, 60, 0, 20],
  },
  body: {
    alignment: 'justify',
    margin: [0, 0, 0, 70],
  },
  signature: {
    fontSize: 14,
    bold: true,
    // alignment: 'left',
  },
  footer: {
    fontSize: 10,
    italics: true,
    alignment: 'center',
    margin: [0, 0, 0, 20],
  },
};

export const getEmploymentLetterByIdReport = (
  values: ReportValues,
): TDocumentDefinitions => {
  const {
    employerName,
    employerPosition,
```

```
    employeeName,
    employeePosition,
    employeeStartDate,
    employeeHours,
    employeeWorkSchedule,
    employerCompany,
  } = values;

  const docDefinition: TDocumentDefinitions = {
    styles: styles,
    pageMargins: [40, 60, 40, 60],

    header: headerSection({
      showLogo: true,
      showDate: true,
    }),

    content: [
      {
        text: 'CONSTANCIA DE EMPLEO',
        style: 'header',
      },
      {
        text: `Yo, ${employerName}, en mi calidad de ${employerPosition} de
${employerCompany}, por medio de la presente certifico que ${employeeName} ha sido
empleado en nuestra empresa desde el
${DateFormatter.getDDMMMMYYYY(employeeStartDate)}. \n\n
        Durante su empleo, el Sr./Sra. ${employeeName} ha desempeñado el cargo de
${employeePosition}, demostrando responsabilidad, compromiso y habilidades
profesionales en sus labores.\n\n
        La jornada laboral del Sr./ Sra. ${employeeName} es de ${employeeHours}
horas semanales, con un horario de ${employeeWorkSchedule}, cumpliendo con las
políticas y procedimientos establecidos por la empresa.\n\n
        Esta constancia se expide a solicitud del interesado para los fines que
considere conveniente. \n\n`,
        style: 'body',
      },
      { text: `Atentamente`, style: 'signature' },
      { text: employerName, style: 'signature' },
      { text: employerPosition, style: 'signature' },
      { text: employerCompany, style: 'signature' },
      { text: DateFormatter.getDDMMMMYYYY(new Date()), style: 'signature' },
    ],

    footer: {
      text: 'Este documento es una constancia de empleo y no representa un
compromiso laboral.',
      style: 'footer',
    },
  };

  return docDefinition;
};
```

NEST REPORTS - TABLA PAISES

- Reutilizar componentes
- Número a pie de página
- Tablas
- Personalización de tablas
- Estilos personalizados reutilizables
- Múltiples tablas en un reporte
- Totales del reporte

Base de dastos de países

- Continuación

> docker compose up -d

- Genero el cliente que tendrá lo necesario para trabajar con la Db definida en .env y el schema

> npx prisma generate

- En localhost:8080 entro en pgAdmin (tengo las credenciales a introducir en el docker-compose)
- Tengo el archivo de creación de la tabla de countries

```
create type public.continents as enum (
    'Africa',
    'Antarctica',
    'Asia',
    'Europe',
    'Oceania',
    'North America',
    'South America'
);
create table public.countries (
  id bigint generated by default as identity primary key,
  name text,
  iso2 text not null,
  iso3 text,
  local_name text,
  continent continents
);
comment on table countries is 'Full list of countries.';
comment on column countries.name is 'Full country name.';
comment on column countries.iso2 is 'ISO 3166-1 alpha-2 code.';
comment on column countries.iso3 is 'ISO 3166-1 alpha-3 code.';
comment on column countries.local_name is 'Local variation of the name.';
insert into public.countries (name,iso2,iso3,local_name,continent) values
  ('Bonaire, Sint Eustatius and Saba','BQ','BES',null,null),
```

```
    ('Curaçao','CW','CUW',null,null),
    ('Guernsey','GG','GGY',null,null),
    ('Isle of Man','IM','IMN',null,null),
    ('Jersey','JE','JEY',null,null),
    ('Åland Islands','AX','ALA',null,null),
    ('Montenegro','ME','MNE',null,null),
    ('Saint Barthélemy','BL','BLM',null,null),
    ('Saint Martin (French part)','MF','MAF',null,null),
    ('Serbia','RS','SRB',null,null),
    ('Sint Maarten (Dutch part)','SX','SXM',null,null),
    ('South Sudan','SS','SSD',null,null),
    ('Timor-Leste','TL','TLS',null,null),
    ('American Samoa','as','ASM','Amerika Samoa','Oceania'),
    ('Andorra','AD','AND','Andorra','Europe'),
    ('Angola','AO','AGO','Angola','Africa'),
    ('Anguilla','AI','AIA','Anguilla','North America'),
    ('Antarctica','AQ','ATA','','Antarctica'),
    ('Antigua and Barbuda','AG','ATG','Antigua and Barbuda','North America'),
    ('Argentina','AR','ARG','Argentina','South America'),
    ('Armenia','AM','ARM','Hajastan','Asia'),
    ('Aruba','AW','ABW','Aruba','North America'),
    ('Australia','AU','AUS','Australia','Oceania'),
    ('Austria','AT','AUT','Österreich','Europe'),
    ('Azerbaijan','AZ','AZE','Azerbaijan','Asia'),
    ('Bahamas','BS','BHS','The Bahamas','North America'),
    ('Bahrain','BH','BHR','Al-Bahrayn','Asia'),
    ('Bangladesh','BD','BGD','Bangladesh','Asia'),
    ('Barbados','BB','BRB','Barbados','North America'),
    ('Belarus','BY','BLR','Belarus','Europe'),
    ('Belgium','BE','BEL','Belgium/Belgique','Europe'),
    ('Belize','BZ','BLZ','Belize','North America'),
    ('Benin','BJ','BEN','Benin','Africa'),
    ('Bermuda','BM','BMU','Bermuda','North America'),
    ('Bhutan','BT','BTN','Druk-Yul','Asia'),
    ('Bolivia','BO','BOL','Bolivia','South America'),
    ('Bosnia and Herzegovina','BA','BIH','Bosna i Hercegovina','Europe'),
    ('Botswana','BW','BWA','Botswana','Africa'),
    ('Bouvet Island','BV','BVT','Bouvet Island','Antarctica'),
    ('Brazil','BR','BRA','Brasil','South America'),
    ('British Indian Ocean Territory','IO','IOT','British Indian Ocean
  Territory','Africa'),
    ('Brunei Darussalam','BN','BRN','Brunei Darussalam','Asia'),
    ('Bulgaria','BG','BGR','Balgarija','Europe'),
    ('Burkina Faso','BF','BFA','Burkina Faso','Africa'),
    ('Burundi','BI','BDI','Burundi/Uburundi','Africa'),
    ('Cambodia','KH','KHM','Cambodia','Asia'),
    ('Cameroon','CM','CMR','Cameroun/Cameroon','Africa'),
    ('Canada','CA','CAN','Canada','North America'),
    ('Cape Verde','CV','CPV','Cabo Verde','Africa'),
    ('Cayman Islands','KY','CYM','Cayman Islands','North America'),
    ('Central African Republic','CF','CAF','Centrafrique','Africa'),
    ('Chad','TD','TCD','Tchad/Tshad','Africa'),
    ('Chile','CL','CHL','Chile','South America'),
    ('China','CN','CHN','Zhongguo','Asia'),
```

```
    ('Christmas Island','CX','CXR','Christmas Island','Oceania'),
    ('Cocos (Keeling) Islands','CC','CCK','Cocos (Keeling) Islands','Oceania'),
    ('Colombia','CO','COL','Colombia','South America'),
    ('Comoros','KM','COM','Komori/Comores','Africa'),
    ('Congo','CG','COG','Congo','Africa'),
    ('Congo, the Democratic Republic of the','CD','COD','Republique Democratique du
  Congo','Africa'),
    ('Cook Islands','CK','COK','The Cook Islands','Oceania'),
    ('Costa Rica','CR','CRI','Costa Rica','North America'),
    ('Cote DIvoire','CI','CIV','Côte dIvoire','Africa'),
    ('Croatia','HR','HRV','Hrvatska','Europe'),
    ('Cuba','CU','CUB','Cuba','North America'),
    ('Cyprus','CY','CYP','Cyprus','Asia'),
    ('Czech Republic','CZ','CZE','Czech','Europe'),
    ('Denmark','DK','DNK','Danmark','Europe'),
    ('Djibouti','DJ','DJI','Djibouti/Jibuti','Africa'),
    ('Dominica','DM','DMA','Dominica','North America'),
    ('Dominican Republic','DO','DOM','Republica Dominicana','North America'),
    ('Ecuador','EC','ECU','Ecuador','South America'),
    ('Egypt','EG','EGY','Misr','Africa'),
    ('El Salvador','SV','SLV','El Salvador','North America'),
    ('Equatorial Guinea','GQ','GNQ','Guinea Ecuatorial','Africa'),
    ('Eritrea','ER','ERI','Ertra','Africa'),
    ('Estonia','EE','EST','Eesti','Europe'),
    ('Ethiopia','ET','ETH','Yeityopiya','Africa'),
    ('Falkland Islands (Malvinas)','FK','FLK','Falkland Islands','South America'),
    ('Faroe Islands','FO','FRO','Faroe Islands','Europe'),
    ('Fiji','FJ','FJI','Fiji Islands','Oceania'),
    ('Finland','FI','FIN','Suomi','Europe'),
    ('France','FR','FRA','France','Europe'),
    ('French Guiana','GF','GUF','Guyane francaise','South America'),
    ('French Polynesia','PF','PYF','Polynésie française','Oceania'),
    ('French Southern Territories','TF','ATF','Terres australes
  françaises','Antarctica'),
    ('Gabon','GA','GAB','Le Gabon','Africa'),
    ('Gambia','GM','GMB','The Gambia','Africa'),
    ('Georgia','GE','GEO','Sakartvelo','Asia'),
    ('Germany','DE','DEU','Deutschland','Europe'),
    ('Ghana','GH','GHA','Ghana','Africa'),
    ('Gibraltar','GI','GIB','Gibraltar','Europe'),
    ('Greece','GR','GRC','Greece','Europe'),
    ('Greenland','GL','GRL','Kalaallit Nunaat','North America'),
    ('Grenada','GD','GRD','Grenada','North America'),
    ('Guadeloupe','GP','GLP','Guadeloupe','North America'),
    ('Guam','GU','GUM','Guam','Oceania'),
    ('Guatemala','GT','GTM','Guatemala','North America'),
    ('Guinea','GN','GIN','Guinea','Africa'),
    ('Guinea-Bissau','GW','GNB','Guinea-Bissau','Africa'),
    ('Guyana','GY','GUY','Guyana','South America'),
    ('Haiti','HT','HTI','Haiti/Dayti','North America'),
    ('Heard Island and Mcdonald Islands','HM','HMD','Heard and McDonald
  Islands','Antarctica'),
    ('Holy See (Vatican City State)','VA','VAT','Santa Sede/Città del
  Vaticano','Europe'),
```

```
    ('Honduras','HN','HND','Honduras','North America'),
    ('Hong Kong','HK','HKG','Xianggang/Hong Kong','Asia'),
    ('Hungary','HU','HUN','Hungary','Europe'),
    ('Iceland','IS','ISL','Iceland','Europe'),
    ('India','IN','IND','Bharat/India','Asia'),
    ('Indonesia','ID','IDN','Indonesia','Asia'),
    ('Iran, Islamic Republic of','IR','IRN','Iran','Asia'),
    ('Iraq','IQ','IRQ','Al-Irāq','Asia'),
    ('Ireland','IE','IRL','Ireland','Europe'),
    ('Israel','IL','ISR','Yisrael','Asia'),
    ('Italy','IT','ITA','Italia','Europe'),
    ('Jamaica','JM','JAM','Jamaica','North America'),
    ('Japan','JP','JPN','Nihon/Nippon','Asia'),
    ('Jordan','JO','JOR','Al-Urdunn','Asia'),
    ('Kazakhstan','KZ','KAZ','Qazaqstan','Asia'),
    ('Kenya','KE','KEN','Kenya','Africa'),
    ('Kiribati','KI','KIR','Kiribati','Oceania'),
    ('Korea, Democratic People''s Republic of','KP','PRK','Choson Minjujuui Inmin
  Konghwaguk (Bukhan)','Asia'),
    ('Korea, Republic of','KR','KOR','Taehan-minguk (Namhan)','Asia'),
    ('Kuwait','KW','KWT','Al-Kuwayt','Asia'),
    ('Kyrgyzstan','KG','KGZ','Kyrgyzstan','Asia'),
    ('Lao People''s Democratic Republic','LA','LAO','Lao','Asia'),
    ('Latvia','LV','LVA','Latvija','Europe'),
    ('Lebanon','LB','LBN','Lubnan','Asia'),
    ('Lesotho','LS','LSO','Lesotho','Africa'),
    ('Liberia','LR','LBR','Liberia','Africa'),
    ('Libya','LY','LBY','Libiya','Africa'),
    ('Liechtenstein','LI','LIE','Liechtenstein','Europe'),
    ('Lithuania','LT','LTU','Lietuva','Europe'),
    ('Luxembourg','LU','LUX','Luxembourg','Europe'),
    ('Macao','MO','MAC','Macau/Aomen','Asia'),
    ('Macedonia, the Former Yugoslav Republic of','MK','MKD','Makedonija','Europe'),
    ('Madagascar','MG','MDG','Madagasikara/Madagascar','Africa'),
    ('Malawi','MW','MWI','Malawi','Africa'),
    ('Malaysia','MY','MYS','Malaysia','Asia'),
    ('Maldives','MV','MDV','Dhivehi Raajje/Maldives','Asia'),
    ('Mali','ML','MLI','Mali','Africa'),
    ('Malta','MT','MLT','Malta','Europe'),
    ('Marshall Islands','MH','MHL','Marshall Islands/Majol','Oceania'),
    ('Martinique','MQ','MTQ','Martinique','North America'),
    ('Mauritania','MR','MRT','Muritaniya/Mauritanie','Africa'),
    ('Mauritius','MU','MUS','Mauritius','Africa'),
    ('Mayotte','YT','MYT','Mayotte','Africa'),
    ('Mexico','MX','MEX','Mexico','North America'),
    ('Micronesia, Federated States of','FM','FSM','Micronesia','Oceania'),
    ('Moldova, Republic of','MD','MDA','Moldova','Europe'),
    ('Monaco','MC','MCO','Monaco','Europe'),
    ('Mongolia','MN','MNG','Mongol Uls','Asia'),
    ('Albania','AL','ALB','Republika e Shqipërisë','Europe'),
    ('Montserrat','MS','MSR','Montserrat','North America'),
    ('Morocco','MA','MAR','Al-Maghrib','Africa'),
    ('Mozambique','MZ','MOZ','Mozambique','Africa'),
    ('Myanmar','MM','MMR','Myanma Pye','Asia'),
```

```
  ('Namibia','NA','NAM','Namibia','Africa'),
  ('Nauru','NR','NRU','Naoero/Nauru','Oceania'),
  ('Nepal','NP','NPL','Nepal','Asia'),
  ('Netherlands','NL','NLD','Nederland','Europe'),
  ('New Caledonia','NC','NCL','Nouvelle-Calédonie','Oceania'),
  ('New Zealand','NZ','NZL','New Zealand/Aotearoa','Oceania'),
  ('Nicaragua','NI','NIC','Nicaragua','North America'),
  ('Niger','NE','NER','Niger','Africa'),
  ('Nigeria','NG','NGA','Nigeria','Africa'),
  ('Niue','NU','NIU','Niue','Oceania'),
  ('Norfolk Island','NF','NFK','Norfolk Island','Oceania'),
  ('Northern Mariana Islands','MP','MNP','Northern Mariana Islands','Oceania'),
  ('Norway','NO','NOR','Norge','Europe'),
  ('Oman','OM','OMN','Oman','Asia'),
  ('Pakistan','PK','PAK','Pakistan','Asia'),
  ('Palau','PW','PLW','Belau/Palau','Oceania'),
  ('Palestine, State of','PS','PSE','Filastin','Asia'),
  ('Panama','PA','PAN','República de Panamá','North America'),
  ('Papua New Guinea','PG','PNG','Papua New Guinea/Papua Niugini','Oceania'),
  ('Paraguay','PY','PRY','Paraguay','South America'),
  ('Peru','PE','PER','Perú/Piruw','South America'),
  ('Philippines','PH','PHL','Pilipinas','Asia'),
  ('Pitcairn','PN','PCN','Pitcairn','Oceania'),
  ('Poland','PL','POL','Polska','Europe'),
  ('Portugal','PT','PRT','Portugal','Europe'),
  ('Puerto Rico','PR','PRI','Puerto Rico','North America'),
  ('Qatar','QA','QAT','Qatar','Asia'),
  ('Reunion','RE','REU','Reunion','Africa'),
  ('Romania','RO','ROM','Romania','Europe'),
  ('Russian Federation','RU','RUS','Rossija','Europe'),
  ('Rwanda','RW','RWA','Rwanda/Urwanda','Africa'),
  ('Saint Helena, Ascension and Tristan da Cunha','SH','SHN','Saint
Helena','Africa'),
  ('Saint Kitts and Nevis','KN','KNA','Saint Kitts and Nevis','North America'),
  ('Saint Lucia','LC','LCA','Saint Lucia','North America'),
  ('Saint Pierre and Miquelon','PM','SPM','Saint-Pierre-et-Miquelon','North
America'),
  ('Saint Vincent and the Grenadines','VC','VCT','Saint Vincent and the
Grenadines','North America'),
  ('Samoa','WS','WSM','Samoa','Oceania'),
  ('San Marino','SM','SMR','San Marino','Europe'),
  ('Sao Tome and Principe','ST','STP','São Tomé e Príncipe','Africa'),
  ('Saudi Arabia','SA','SAU','Al-Mamlaka al-Arabiya as-Saudiya','Asia'),
  ('Senegal','SN','SEN','Sénégal/Sounougal','Africa'),
  ('Seychelles','SC','SYC','Sesel/Seychelles','Africa'),
  ('Sierra Leone','SL','SLE','Sierra Leone','Africa'),
  ('Singapore','SG','SGP','Singapore/Singapura/Xinjiapo/Singapur','Asia'),
  ('Slovakia','SK','SVK','Slovensko','Europe'),
  ('Slovenia','SI','SVN','Slovenija','Europe'),
  ('Solomon Islands','SB','SLB','Solomon Islands','Oceania'),
  ('Somalia','SO','SOM','Soomaaliya','Africa'),
  ('South Africa','ZA','ZAF','South Africa','Africa'),
  ('South Georgia and the South Sandwich Islands','GS','SGS','South Georgia and
the South Sandwich Islands','Antarctica'),
```

```
  ('Spain','ES','ESP','España','Europe'),
  ('Sri Lanka','LK','LKA','Sri Lanka/Ilankai','Asia'),
  ('Sudan','SD','SDN','As-Sudan','Africa'),
  ('Suriname','SR','SUR','Suriname','South America'),
  ('Svalbard and Jan Mayen','SJ','SJM','Svalbard og Jan Mayen','Europe'),
  ('Swaziland','SZ','SWZ','kaNgwane','Africa'),
  ('Sweden','SE','SWE','Sverige','Europe'),
  ('Switzerland','CH','CHE','Schweiz/Suisse/Svizzera/Svizra','Europe'),
  ('Syrian Arab Republic','SY','SYR','Suriya','Asia'),
  ('Taiwan (Province of China)','TW','TWN','Tai-wan','Asia'),
  ('Tajikistan','TJ','TJK','Tajikistan','Asia'),
  ('Tanzania, United Republic of','TZ','TZA','Tanzania','Africa'),
  ('Thailand','TH','THA','Prathet Thai','Asia'),
  ('Togo','TG','TGO','Togo','Africa'),
  ('Tokelau','TK','TKL','Tokelau','Oceania'),
  ('Tonga','TO','TON','Tonga','Oceania'),
  ('Trinidad and Tobago','TT','TTO','Trinidad and Tobago','North America'),
  ('Tunisia','TN','TUN','Tunis/Tunisie','Africa'),
  ('Turkey','TR','TUR','Türkiye','Asia'),
  ('Turkmenistan','TM','TKM','Türkmenistan','Asia'),
  ('Turks and Caicos Islands','TC','TCA','The Turks and Caicos Islands','North
America'),
  ('Tuvalu','TV','TUV','Tuvalu','Oceania'),
  ('Uganda','UG','UGA','Uganda','Africa'),
  ('Ukraine','UA','UKR','Ukrajina','Europe'),
  ('United Arab Emirates','AE','ARE','Al-Amirat al-Arabiya al-Muttahida','Asia'),
  ('United Kingdom','GB','GBR','United Kingdom','Europe'),
  ('United States','US','USA','United States','North America'),
  ('United States Minor Outlying Islands','UM','UMI','United States Minor Outlying
Islands','Oceania'),
  ('Uruguay','UY','URY','Uruguay','South America'),
  ('Uzbekistan','UZ','UZB','Uzbekiston','Asia'),
  ('Vanuatu','VU','VUT','Vanuatu','Oceania'),
  ('Venezuela','VE','VEN','Venezuela','South America'),
  ('Viet Nam','VN','VNM','Viet Nam','Asia'),
  ('Virgin Islands (British)','VG','VGB','British Virgin Islands','North
America'),
  ('Virgin Islands (U.S.)','VI','VIR','Virgin Islands of the United States','North
America'),
  ('Wallis and Futuna','WF','WLF','Wallis-et-Futuna','Oceania'),
  ('Western Sahara','EH','ESH','As-Sahrawiya','Africa'),
  ('Yemen','YE','YEM','Al-Yaman','Asia'),
  ('Zambia','ZM','ZMB','Zambia','Africa'),
  ('Zimbabwe','ZW','ZWE','Zimbabwe','Africa'),
  ('Afghanistan','AF','AFG','Afganistan/Afqanestan','Asia'),
  ('Algeria','DZ','DZA','Al-Jazair/Algerie','Africa');
```

- Luego haremos tablas relacionadas
- En pgAdmin , a la izquierda, en databases, selecciono postgres
- Clico en el icono de DB Query Tools, me aseguro que en la barra tengo postgres/postgres@Reports Database y pego el código de aqui arriba
- Si hago una consulta

```
select * from countries
```

- Hay registros que tienen el apartado local_name en NULL
- Lo purgaremos mediante un query
- Lo puedo querer ordenar alfabéticamente, por continente, lo que sea
- Lo podemos hacer
- Para obtener los schemas hacemos un **npx prisma db pull**
- prisma.schema

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model employees {
  id            Int      @id @default(autoincrement())
  name          String   @db.VarChar(100)
  position      String   @db.VarChar(50)
  start_date    DateTime @db.Date
  work_time     DateTime @db.Time(6)
  hours_per_day Int
  work_schedule String   @db.VarChar(50)
}

/// This model or at least one of its fields has comments in the database, and
requires an additional setup for migrations: Read more:
https://pris.ly/d/database-comments
model countries {
  id         BigInt      @id @default(autoincrement())
  name       String?
  iso2       String
  iso3       String?
  local_name String?
  continent  continents?
}

model categories {
  category_id   Int        @id @default(autoincrement())
  category_name String?    @db.VarChar(255)
  description   String?    @db.VarChar(255)
  products      products[]
}

model customers {
  customer_id   Int      @id @default(autoincrement())
```

```
  customer_name String?  @db.VarChar(255)
  contact_name  String?  @db.VarChar(255)
  address       String?  @db.VarChar(255)
  city          String?  @db.VarChar(255)
  postal_code   String?  @db.VarChar(255)
  country       String?  @db.VarChar(255)
  orders        orders[]
}

model order_details {
  order_detail_id Int       @id @default(autoincrement())
  order_id        Int?
  product_id      Int?
  quantity        Int?
  orders          orders?   @relation(fields: [order_id], references: [order_id],
onDelete: NoAction, onUpdate: NoAction)
  products        products? @relation(fields: [product_id], references:
[product_id], onDelete: NoAction, onUpdate: NoAction)
}

model orders {
  order_id      Int             @id @default(autoincrement())
  customer_id   Int?
  order_date    DateTime?       @db.Date
  order_details order_details[]
  customers     customers?      @relation(fields: [customer_id], references:
[customer_id], onDelete: NoAction, onUpdate: NoAction)
}

model products {
  product_id    Int             @id @default(autoincrement())
  product_name  String?         @db.VarChar(255)
  category_id   Int?
  unit          String?         @db.VarChar(255)
  price         Decimal?        @db.Decimal(10, 2)
  order_details order_details[]
  categories    categories?     @relation(fields: [category_id], references:
[category_id], onDelete: NoAction, onUpdate: NoAction)
}

enum continents {
  Africa
  Antarctica
  Asia
  Europe
  Oceania
  North_America @map("North America")
  South_America @map("South America")
}
```

- Ahora puedo ejecutar **npx prisma generate** para generar el cliente que disponga de los schemas para interactuar con la DB

Reportes con tablas

- En reports-download tengo el documento pdf final para observar (descargado)
- Vamos con el controlador

```
  //countries
  @Get('countries')
  async getCountriesReport(@Res() response: Response) {
    const pdfDoc = await this.basicReportsService.getCountries();

    response.setHeader('Content-Type', 'application/pdf');
    pdfDoc.info.Title = 'Countries-Report';
    pdfDoc.pipe(response);
    pdfDoc.end();
  }
```

- En el servicio

```
async getCountries() {
const countries = await this.countries.findMany({
    where: {
    local_name: {
        not: null,
    },
    },
});

const docDefinition = getCountryReport({ countries });

return this.printerService.createPdf(docDefinition);
}
```

- El caso de uso de reports/country.report
- Renombro countries de prisma/client a Country. Dispongo gracias al cliente y los schemas de las tablas que he creado como objetos nativamente
- Si tengo un titulo en las opciones lo añado, si no le añado el string que he puesto como título
- Lo mismo con el subtítulo
- El footerSection lo importo de sections
- En el arreglo de content, especifico el tipo de layout
- headerRows
- Con widths indico la anchura de las columnas, * distribuye de manera equitativa y llena el espacio disponible, auto establece el width según el contenido
- Si tenemos 5 columnas debemos espercificar 5 valores. El valor por defecto es auto
- En el body indico en un arreglo la cabecera de las columnas
- Esparzo el countries.map pasándole en orden las propiedades
- Le paso un arreglo con los campos (columnas) en blanco

- Para los totales genero una nueva tabla pero sin borders
- Coloco colSpan en 2, para indicar que esta columna toma 2 posiciones.
- Para que surja efecto y quede bien, recreo el widths (las columnas) de la tabla y coloco los objetos en blanco de las columnas restantes

```
import { TDocumentDefinitions } from 'pdfmake/interfaces';
import { headerSection } from './sections/header.section';
import { countries as Country } from '@prisma/client';
import { footerSection } from './sections/footer.section';

interface ReportOptions {
  title?: string;
  subTitle?: string;
  countries: Country[]; //arreglo de Country (mi schema)
}

export const getCountryReport = (
  options: ReportOptions,
): TDocumentDefinitions => {
  const { title, subTitle, countries } = options;

  return {
    pageOrientation: 'landscape', //horizontal
    header: headerSection({
      title: title ?? 'Countries Report',
      subTitle: subTitle ?? 'List of countries',
    }),
    footer: footerSection,
    pageMargins: [40, 110, 40, 60],
    content: [
      {
        layout: 'customLayout01', //'lightHorizontalLines', // optional
        table: {
          // headers are automatically repeated if the table spans over multiple
pages
          // you can declare how many rows should be treated as headers
          headerRows: 1, //una fila de header
          widths: [50, 50, 50, '*', 'auto', '*'],

          body: [
            ['ID', 'ISO2', 'ISO3', 'Name', 'Continent', 'Local Name'],
            ...countries.map((country) => [
              country.id.toString(),
              country.iso2,
              country.iso3,
              { text: country.name, bold: true },
              country.continent,
              country.local_name,
            ]),

            ['', '', '', '', '', ``], //Dejo una fila en blanco (espacio) al final
            [
```

```
              '',
              '',
              '',
              '',
              'Total',
              {
                text: `${countries.length} países`,
                bold: true,
              },
            ],

            // [{ text: 'Bold value', bold: true }, 'Val 2', 'Val 3', 'Val 4'],
          ],
        },
      },

      // Tabla de totales
      {
        text: 'Totales',
        style: {
          fontSize: 18,
          bold: true,
          margin: [0, 40, 0, 0],
        },
      },
      {
        layout: 'noBorders', // creo una tabla sin lineas
        table: {
          headerRows: 1, //solo un row
          widths: [50, 50, 70, '*', 'auto', '*'], //recreo los mismos widths de la
tabla
          //el body de una tabla siempre es un arreglo, por lo que es un arreglo
de objetos dentro de otro arreglo
          body: [
            [
              {
                text: 'Total de países',
                colSpan: 2, //uso el span para adecuar los espacios
                bold: true,
              },
              {},
              {
                text: `${countries.length} países`,
                bold: true,
              },
              {}, //los espacios en blanco que corresponden a las columnas de los
widths que he recreado
              {},
              {},
            ],
          ],
        },
      },
    ],
```

```
  };
};
```

- Para trabajar el header section como una columna, en lugar del text usaremos **stack**
- Los componentes que estoy creando son de tipo Content
- Al colocarle 150 en el width de CurrentDate, indico que siempre tendrá un tamaño específico
- La columna de headerTitle que no tiene tamaño específico va a tomar el tamaño que necesite

```
import { Content } from 'pdfmake/interfaces';
import { DateFormatter } from 'src/helpers';

const logo: Content = {
  image: 'src/assets/tucan-code-logo.png',
  width: 100,
  height: 100,
  alignment: 'center',
  margin: [0, 0, 0, 20],
};

const currentDate: Content = {
  text: DateFormatter.getDDMMMMYYYY(new Date()),
  alignment: 'right',
  margin: [20, 30],
  width: 150,
};

interface HeaderOptions {
  title?: string;
  subTitle?: string;
  showLogo?: boolean;
  showDate?: boolean;
}

export const headerSection = (options: HeaderOptions): Content => {
  const { title, subTitle, showLogo = true, showDate = true } = options;

  const headerLogo: Content = showLogo ? logo : null;
  const headerDate: Content = showDate ? currentDate : null;

  const headerSubTitle: Content = subTitle
    ? {
        text: subTitle,
        alignment: 'center',
        margin: [0, 2, 0, 0],
        style: {
          fontSize: 16,
          bold: true,
        },
      }
    : null;
```

```
  const headerTitle: Content = title
    ? {
        stack: [
          {
            text: title,
            alignment: 'center',
            margin: [0, 15, 0, 0],
            style: {
              bold: true,
              fontSize: 22,
            },
          },
          headerSubTitle,
        ],
        // text: title,
        // style: {
        //   bold: true,
        // },
      }
    : null;

  return {
    columns: [headerLogo, headerTitle, headerDate],
  };
};
```

- section/footer.section

```
import { Content, ContextPageSize } from 'pdfmake/interfaces';

export const footerSection = (
  currentPage: number,
  pageCount: number,
  pageSize: ContextPageSize,
): Content => {
  return {
    text: `Página ${currentPage} de ${pageCount}`,
    alignment: 'right',
    fontSize: 12,
    bold: true,
    margin: [0, 10, 35, 0],
  };
};
```

- CuurencyFormatter

```
export class CurrencyFormatter {
  static formatCurrency(value: number): string {
    return new Intl.NumberFormat('en-US', {
```

```
      style: 'currency',
      currency: 'USD',
    }).format(value);
  }
}
```

Estilo personalizado de las tablas

- Podemos crear nuestors propios layouts
- Las configuración es estraña. Es cuando se configura la impresora
- Se hace en el objeto de opciones que le mandamos a la impresora
- En el tipo de retorno
- uso Record, sirve para tipar llaves dinámicas que pueden ser cualquier cantidad de elementos
- La llave será un string y el valor un CustomTableLayout
- Lo llamo customLayout01
- Para definir un layout necesito definir las opciones que vienen en el ejemplo de la documentación (este)
- En src/printer/printer.service hago un copy past de la documentación

```
const customTableLayouts: Record<string, CustomTableLayout> = {
  customLayout01: {
    hLineWidth: function (i, node) {
      if (i === 0 || i === node.table.body.length) {
        return 0;
      }
      return i === node.table.headerRows ? 2 : 1;
    },
    vLineWidth: function () {
      return 0;
    },
    hLineColor: function (i) {
      return i === 1 ? 'black' : '#bbbbbb'; //color lineas de la tabla
    },
    paddingLeft: function (i) {
      return i === 0 ? 0 : 8;
    },
    paddingRight: function (i, node) {
      return i === node.table.widths.length - 1 ? 0 : 8;
    },
    fillColor: function (i, node) {
      if (i === 0) {
        return '#7b90be';
      }
      if (i === node.table.body.length - 1) {
        return '#acb3c1';
      }

      return i % 2 === 0 ? '#f3f3f3' : null; //intercalado de colores por pares e
impares
```

```
    },
  },
};
```

- Necesito definirlo cuando defino el printer
- En el printerService

```
@Injectable()
export class PrinterService {
  private printer = new PdfPrinter(fonts);

  createPdf(
    docDefinition: TDocumentDefinitions,
    options: BufferOptions = {
      tableLayouts: customTableLayouts,
    },
  ): PDFKit.PDFDocument {
    return this.printer.createPdfKitDocument(docDefinition, options);
  }
}
```

---

04 NEST REPORTS - MAESTRO DETALLE

- Veremos
  - Código QR
  - Inner joins
  - Tablas y estilos
  - Estructura y alineamiento
  - Envío de datos de Prisma a reporte

- Necesitamos establecer una relación entre ORDER_DETAILS PRODUCTS y COSTUMERS
- Creo el módulo de satoreReports con **nest g resource storeReports --no-spec**
- Necesito el prinerModule para usar el servicio de la impresora
  - El servicio ya está excportado de PrinterModule, importo el módulo en StoreReportsModule

```
import { Module } from '@nestjs/common';
import { StoreReportsService } from './store-reports.service';
import { StoreReportsController } from './store-reports.controller';
import { PrinterModule } from 'src/printer/printer.module';

@Module({
  controllers: [StoreReportsController],
  providers: [StoreReportsService],
  imports: [PrinterModule],
```

```
})
export class StoreReportsModule {}
```

- En el StorereportsController hago lo mismo que en el caso anterior
  - Tomo la Response con @Res
  - Pido el id por @Params
  - En el servicio inyecto el PrinterService, creo el doc
  - Seteo los headers
  - Enlazo la response con el doc usando pipe
  - Cierro la conexión

```
import { Controller, Get, Param, Res } from '@nestjs/common';
import { StoreReportsService } from './store-reports.service';
import { Response } from 'express';

@Controller('store-reports')
export class StoreReportsController {
  constructor(private readonly storeReportsService: StoreReportsService) {}

  @Get('orders/:orderId')
  async getOrderReport(
    @Res() response: Response,
    @Param('orderId') orderId: string,
  ) {
    const pdfDoc = await this.storeReportsService.getOrderByIdReport(+orderId);

    response.setHeader('Content-Type', 'application/pdf');
    pdfDoc.info.Title = 'Order-Report';
    pdfDoc.pipe(response);
    pdfDoc.end();
  }
}
```

- En StoreReportsService busco por id con prisma, incluyo el campo costumers y el campo products
- Debo crear estas tablas y estas relaciones primero
- **NOTA**: CÓDIGO SPOILER!! Todavía no se han hecho estas tablas ni las relaciones

```
import { Injectable, NotFoundException, OnModuleInit } from '@nestjs/common';
import { PrismaClient } from '@prisma/client';
import { PrinterService } from 'src/printer/printer.service';
import { orderByIdReport } from 'src/reports';

@Injectable()
export class StoreReportsService extends PrismaClient implements OnModuleInit {
  async onModuleInit() {
    await this.$connect();
    // console.log('Connected to the database');
  }
```

```
  constructor(private readonly printerService: PrinterService) {
    super(); //usando primsa tengo que llamar al constructor padre
  }

  async getOrderByIdReport(orderId: number) {
    const order = await this.orders.findUnique({
      where: {
        order_id: orderId,
      },
      include: {
        customers: true,
        order_details: {
          include: {
            products: true,
          },
        },
      },
    });

    if (!order) {
      throw new NotFoundException(`Order with id ${orderId} not found`);
    }

    const docDefinition = orderByIdReport({
      data: order as any,
    });

    const doc = this.printerService.createPdf(docDefinition);

    return doc;
  }
}
```

Creación del reporte

- En src/reports/orderById.reports.ts creo varias interfaces
- Piensa en la composición del documento como filas horizontales (y verticales, si delimitas columnas)

```
import type {
  Content,
  StyleDictionary,
  TDocumentDefinitions,
} from 'pdfmake/interfaces';
import { CurrencyFormatter, DateFormatter } from 'src/helpers';
import { footerSection } from './sections/footer.section';

const logo: Content = {
  image: 'src/assets/tucan-banner.png',
  width: 100,
```

```
  height: 30,
  margin: [10, 30],
};

const styles: StyleDictionary = {
  header: {
    fontSize: 20,
    bold: true,
    margin: [0, 30, 0, 0],
  },
  subHeader: {
    fontSize: 16,
    bold: true,
    margin: [0, 20, 0, 0],
  },
};

// esta interfaz sale de Paste JSON as Code con el resultado de un objeto vacío
del método serializado con JSON.stringify
export interface CompleteOrder {
  order_id: number;
  customer_id: number;
  order_date: Date;
  customers: Customers;
  order_details: OrderDetail[];
}

export interface Customers {
  customer_id: number;
  customer_name: string;
  contact_name: string;
  address: string;
  city: string;
  postal_code: string;
  country: string;
}

export interface OrderDetail {
  order_detail_id: number;
  order_id: number;
  product_id: number;
  quantity: number;
  products: Products;
}

export interface Products {
  product_id: number;
  product_name: string;
  category_id: number;
  unit: string;
  price: string;
}

interface ReportValues {
```

```
  title?: string;
  subTitle?: string;
  data: CompleteOrder; //para sacar esta interfaz he ejecutado order=
orderByIdReport({}) con un objeto vacío
                       //y he hecho un console.log(JSON.stringify(order, null, 2))
que me da un JSON válido
                       //copio la respuesta a Paste JSON as Code para que me de las
interfaces
}

export const orderByIdReport = (value: ReportValues): TDocumentDefinitions => {
  const { data } = value;

  const { customers, order_details } = data;

//uso un reduce para obtener el total
  const subTotal = order_details.reduce(
    (acc, detail) => acc + detail.quantity * +detail.products.price,
    0,
  );

  const total = subTotal * 1.15; //Le sumo el 15% de IVA

  return {
    styles: styles,
    header: logo,
    pageMargins: [40, 60, 40, 60],
    footer: footerSection,
    content: [
      // Headers
      {
        text: 'Tucan Code',
        style: 'header',
      },

      // Address y número recibo
      {
        columns: [
          {
            text: '15 Montgomery Str, Suite 100, \nOttawa ON K2Y 9X1, CANADA\nBN:
12783671823\nhttps://devtalles.com',
          },
          { //para aplicar estilos en la misma linea puedo encerrar el texto entre
corchetes y aplicar style:, o bold:true
            text: [
              {
                text: `Recibo No. ${data.order_id}\n`,
                bold: true,
              },
              `Fecha del recibo
${DateFormatter.getDDMMMMYYYY(data.order_date)}\nPagar antes de:
${DateFormatter.getDDMMMMYYYY(new Date())}\n`,
            ],
            alignment: 'right',
```

```
          },
        ],
      },

      // QR   CON SOLO ESTO YA CREO EL QR, fit para el tamaño
      { qr: 'https://devtalles.com', fit: 75, alignment: 'right' },

      // Dirección del cliente
      {
        text: [
          {
            text: 'Cobrar a: \n',
            style: 'subHeader',
          },
          `Razón Social: ${customers.customer_name},
          Contacto: ${customers.contact_name}`,
        ],
      },

      // Table del detalle de la orden
      {
        layout: 'headerLineOnly',
        margin: [0, 20],
        table: {
          headerRows: 1,
          widths: [50, '*', 'auto', 'auto', 'auto'], //5 columnas
          body: [
            ['ID', 'Descripción', 'Cantidad', 'Precio', 'Total'], //Las 5 columnas

            ...order_details.map((detail) => [
              detail.order_detail_id.toString(),
              detail.products.product_name,
              detail.quantity.toString(),
              {
                text: CurrencyFormatter.formatCurrency(+detail.products.price),
                alignment: 'right',
              },
              {
                text: CurrencyFormatter.formatCurrency(
                  +detail.products.price * detail.quantity,
                ),
                alignment: 'right',
              },
            ]),
          ],
        },
      },

      // Salto de línea
      '\n',

      // Totales
      {
        columns: [
```

```
          {
            width: '*',
            text: '',
          },
          {
            width: 'auto',
            layout: 'noBorders',
            table: {
              body: [
                [
                  'Subtotal',
                  {
                    text: CurrencyFormatter.formatCurrency(subTotal),
                    alignment: 'right',
                  },
                ],
                [
                  { text: 'Total', bold: true },
                  {
                    text: CurrencyFormatter.formatCurrency(total),
                    alignment: 'right',
                    bold: true,
                  },
                ],
              ],
            },
          },
        ],
      },
    ],
  };
};
```

- En el schema puedo crear los modelos manualmente o hacerlo con un pull después de crear las tablas en un query

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model employees {
  id            Int      @id @default(autoincrement())
  name          String   @db.VarChar(100)
  position      String   @db.VarChar(50)
  start_date    DateTime @db.Date
  work_time     DateTime @db.Time(6)
```

```
  hours_per_day Int
  work_schedule String   @db.VarChar(50)
}

/// This model or at least one of its fields has comments in the database, and
requires an additional setup for migrations: Read more:
https://pris.ly/d/database-comments
model countries {
  id         BigInt      @id @default(autoincrement())
  name       String?
  iso2       String
  iso3       String?
  local_name String?
  continent  continents?
}

model categories {
  category_id   Int        @id @default(autoincrement())
  category_name String?    @db.VarChar(255)
  description   String?    @db.VarChar(255)
  products      products[]
}

model customers {
  customer_id   Int      @id @default(autoincrement())
  customer_name String?  @db.VarChar(255)
  contact_name  String?  @db.VarChar(255)
  address       String?  @db.VarChar(255)
  city          String?  @db.VarChar(255)
  postal_code   String?  @db.VarChar(255)
  country       String?  @db.VarChar(255)
  orders        orders[]
}

model order_details {
  order_detail_id Int       @id @default(autoincrement())
  order_id        Int?
  product_id      Int?
  quantity        Int?
  orders          orders?   @relation(fields: [order_id], references: [order_id],
onDelete: NoAction, onUpdate: NoAction)
  products        products? @relation(fields: [product_id], references:
[product_id], onDelete: NoAction, onUpdate: NoAction)
}

model orders {
  order_id      Int             @id @default(autoincrement())
  customer_id   Int?
  order_date    DateTime?       @db.Date
  order_details order_details[]
  customers     customers?      @relation(fields: [customer_id], references:
[customer_id], onDelete: NoAction, onUpdate: NoAction)
}
```

```
model products {
  product_id    Int             @id @default(autoincrement())
  product_name  String?         @db.VarChar(255)
  category_id   Int?
  unit          String?         @db.VarChar(255)
  price         Decimal?        @db.Decimal(10, 2)
  order_details order_details[]
  categories    categories?     @relation(fields: [category_id], references:
[category_id], onDelete: NoAction, onUpdate: NoAction)
}

enum continents {
  Africa
  Antarctica
  Asia
  Europe
  Oceania
  North_America @map("North America")
  South_America @map("South America")
}
```

Código QR

- Crear el código QR es sencillo
- Tan fácil como escribir

```
const docDefinition={
  content:[
    {qr: "Texto dentro del qr"}
  ]

}
```

Relaciones de las tablas

- Cómo sería la relación para la orden 10250

```
SELECT
*
FROM
    ORDERS
    INNER JOIN  ORDER_DETAILS ON ORDERS.ORDER_ID = ORDER_DETAILS.ORDER.ID
WHERE
    ORDERS.ORDER_id = 10250
```

- DE ESTA MANERA SABEMOS CUANTOS PRODUCTOS SE LLEVA PERO NO TENEMOS LA INFO DEL PRODUCTO

```
SELECT
*
FROM
    ORDERS
    INNER JOIN  ORDER_DETAILS ON ORDERS.ORDER_ID = ORDER_DETAILS.ORDER.ID
    INNER JOIN products on  ORDER_DETAILS.product_id = products.product_id
WHERE
    ORDERS.ORDER_id = 10250
```

- De esta manera ya tenemos el nombre del producto, la categoría, el precio...
- Necesitamos otro INNER JOIN para el cliente

```
SELECT
*
FROM
    ORDERS
    INNER JOIN  ORDER_DETAILS ON ORDERS.ORDER_ID = ORDER_DETAILS.ORDER.ID
    INNER JOIN products on  ORDER_DETAILS.product_id = products.product_id
    INNER JOIN customers on orders.customer.id = customers.customer_id
WHERE
    ORDERS.ORDER_id = 10250
```

---

falta trabajo con gráficos