



Proposta de Trabalho Prático 2

Integração de Sistemas de Informação

21110 – Flávio Pereira

Licenciatura em Engenharia de Sistemas Informáticos
3ºano

Barcelos | Dezembro, 2023

Conteúdo

| | |
|---|----|
| Introdução..... | 3 |
| Processo de Documentação..... | 4 |
| Documentação Criada pelo Doxygen..... | 5 |
| Visão Geral do Projeto | 6 |
| ETL..... | 6 |
| Serviços de dados Covid..... | 7 |
| Serviços de dados de Geolocalização..... | 12 |
| Demonstração dos serviços dados Covid e Geolocalização..... | 14 |
| Serviços de Autenticação | 15 |
| Demonstração serviços de Autenticação..... | 19 |
| Dashboard..... | 20 |
| Publicação na Cloud | 25 |
| Conclusão | 26 |

Introdução

No decorrer deste projeto, foi utilizada uma extensa quantidade de dados coletados durante a pandemia de Covid-19 em diversos países da Europa, abrangendo o período de 2020 a 2022. O foco principal foi direcionado aos dados referentes ao ano de 2022, complementando-os com um serviço de geolocalização. Essa abordagem visa proporcionar uma compreensão mais aprofundada sobre a evolução da pandemia ao longo desse período, permitindo uma representação visual dos dados, por exemplo, em um mapa.

Esta etapa do projeto se integra à Fase 1, na qual foi demonstrada a utilidade das ferramentas de ETL(Extract, transform, load) e solidificados os conceitos de integração de sistemas. Foram utilizados serviços web e integradas APIs externas, além da apresentação dos serviços por meio de um painel e disponibilização na nuvem (Computação em nuvem).

Dessa forma, o projeto não apenas explora a manipulação e análise de dados relevantes da pandemia, mas também destaca a aplicação prática de ferramentas de integração, evidenciando a utilidade e eficácia desses serviços em um contexto mais amplo.

Processo de Documentação

O processo de documentação foi realizado através do [Doxygen](#) e [Graphviz](#), uma ferramenta eficiente para gerar documentação a partir de código-fonte. A escolha do Doxygen baseou-se na sua capacidade de analisar automaticamente o código e criar documentação consistente e bem formatada.

Esse processo consistiu nos seguintes passos:

1. **Integração do Doxygen:** O Doxygen foi integrado ao projeto para analisar o código-fonte automaticamente.
2. **Configuração do Doxyfile:** Um ficheiro de configuração (Doxyfile) foi criado para especificar as configurações desejadas para a criação da documentação.
3. **Marcação do Código:** O código-fonte foi marcado com comentários especiais Doxygen para descrever classes, métodos, parâmetros e outros elementos significativos.
4. **Execução do Doxygen:** O Doxygen foi executado para processar os comentários marcados e criar documentação em diferentes formatos como [HTML](#) e [LaTeX](#) (que foi usado para criar um ficheiro em formato PDF), utilizando também o Graphviz para gerar gráfico.

Documentação Criada pelo Doxygen

Documentação HTML: A documentação em HTML pode ser acessada na pasta HTML criada pelo Doxygen (presente dentro da pasta de projeto) onde a documentação foi no ficheiro "index.html". A documentação em HTML é mais amigável e interativa, permitindo uma procura mais fácil pelas diferentes seções do projeto.

Documentação em PDF: Para a versão em PDF, **foi utilizado** o [MiKTeX](#) para instalar os componentes necessários e o [TeXworks](#) para compilar o ficheiro LaTeX gerado pelo Doxygen. O ficheiro PDF está incluído na pasta do projeto. A documentação em PDF fornece uma versão estática e portátil da documentação.

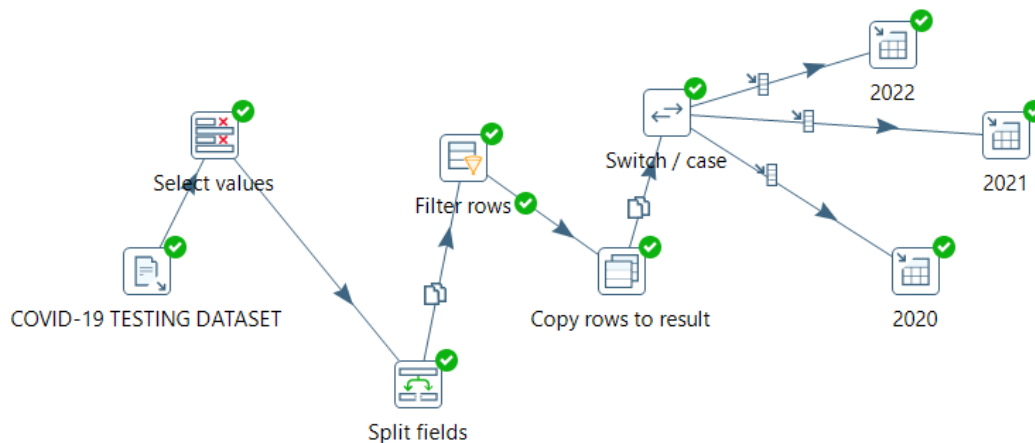
Visão Geral do Projeto

Este projeto consiste em uma REST API destinada à gestão e análise de dados relacionados à pandemia de Covid-19 para além da criação de uma Dashboard para utilização da mesma. Abaixo, destacam-se os principais componentes e funcionalidades do projeto:

ETL

Inicialmente foi utilizado um CSV [Dataset](#) que incluiu informações relativas à testagem da Covid-19 na época da pandemia (2020-2022), que continha dados convenientes, tais como, País, Região, Ano-Semana, Nivel de Testagem, Novos Casos, Testes Efetuados, População, Taxa de Positividade e Origem da Testagem. Foi utilizado o MySQL para guardar dados relativos aos anos numa base de dados.

A primeira transformação recebeu o ficheiro CSV como data input, contudo os dados neste dataset precisavam de ser limpos e estruturados, então foram efetuadas uma série de operações de Seleção, Separação e Filtragem para alterar o desejado.



Serviços de dados Covid

Foram criados dois modelos, primeiramente a classe [CovidAPI.Models.CovidData](#) representa dados da COVID-19 armazenados na base de dados. Inclui as seguintes propriedades:

| | |
|---|---|
| int Id [get, set] | Gets or sets the unique identifier for the Covid-19 data. |
| string? Country [get, set] | Gets or sets the name of the country where the data belongs. |
| string? CountryCode [get, set] | Gets or sets the country code for the data. |
| int Year [get, set] | Gets or sets the year for which the data is recorded. |
| string? Week [get, set] | Gets or sets the week within the year for which the data is recorded. |
| string? Region [get, set] | Gets or sets the region where the data is recorded. |
| string? RegionName [get, set] | Gets or sets the name of the region where the data is recorded. |
| int NewCases [get, set] | Gets or sets the number of new cases reported. |
| int TestsDone [get, set] | Gets or sets the number of tests conducted. |
| int Population [get, set] | Gets or sets the population of the region. |
| double TestingRate [get, set] | Gets or sets the rate of testing. |
| double PositivityRate [get, set] | Gets or sets the positivity rate of the tests conducted. |
| string? TestingDataSource [get, set] | Gets or sets the source of testing data. |

CovidAPI.Models.CovidData

```
+ Id
+ Country
+ CountryCode
+ Year
+ Week
+ Region
+ RegionName
+ NewCases
+ TestsDone
+ Population
+ TestingRate
+ PositivityRate
+ TestingDataSource
```

E a classe [*CovidAPI.Models.CovidDataDTO*](#) representa dados da COVID-19 armazenados na base de dados. Inclui as seguintes propriedades:

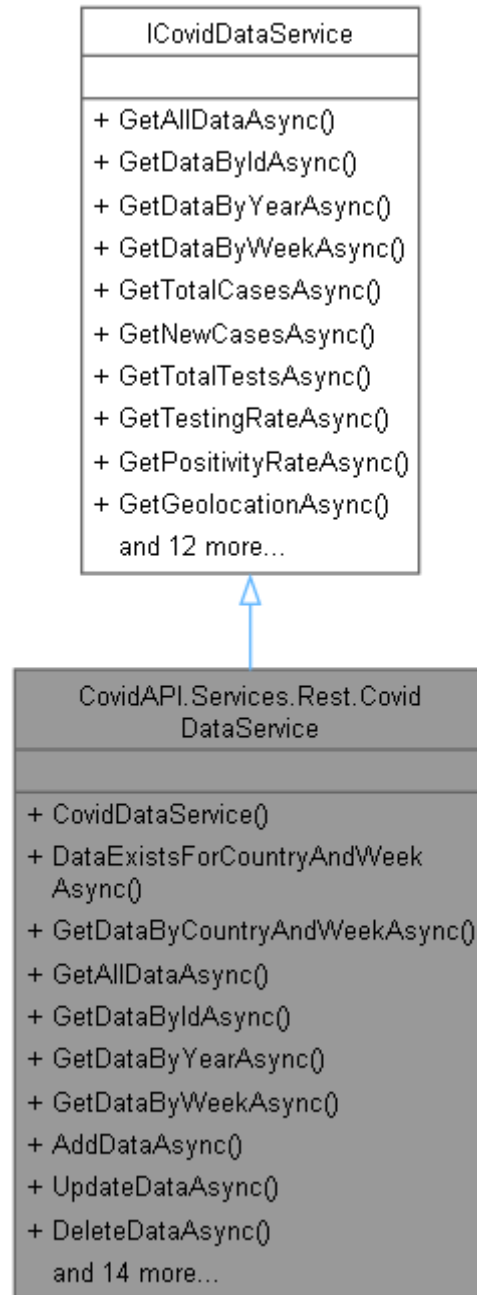
| | | |
|------------------------|-------------------------------------|--|
| int | Id [get, set] | Gets or sets the unique identifier for the Covid-19 data. |
| string | Country [get, set] | Gets or sets the name of the country where the data belongs. |
| string | CountryCode [get, set] | Gets or sets the country code for the data. |
| int | Year [get, set] | Gets or sets the year for which the data is recorded. |
| string | Week [get, set] | Gets or sets the week within the year for which the data is recorded. |
| string | Region [get, set] | Gets or sets the region where the data is recorded. |
| string | RegionName [get, set] | Gets or sets the name of the region where the data is recorded. |
| int | NewCases [get, set] | Gets or sets the number of new cases reported. |
| int | TestsDone [get, set] | Gets or sets the number of tests conducted. |
| int | Population [get, set] | Gets or sets the population of the region. |
| double | PositivityRate [get, set] | Gets or sets the positivity rate of the tests conducted. |
| double | TestingRate [get, set] | Gets or sets the rate of testing. |
| string | TestingDataSource [get, set] | Gets or sets the source of testing data. |
| int? | TotalCasesYear [get, set] | Gets or sets the total cases reported in a specific year. |
| int? | TotalTestsYear [get, set] | Gets or sets the total tests conducted in a specific year. |
| double? | PerCapitaCases [get, set] | Gets or sets the per capita cases (cases per population) for the data. |
| double? | PerCapitaTests [get, set] | Gets or sets the per capita tests (tests per population) for the data. |
| GeolocationComponents? | Geolocation [get, set] | Gets or sets the geolocation information for the data. |
| Geometry? | Geometry [get, set] | Gets or sets the geometric information for the data. |

| CovidAPI.Models.CovidDataDTO | |
|------------------------------|---------------|
| + | Id |
| + | Country |
| + | CountryCode |
| + | Year |
| + | Week |
| + | Region |
| + | RegionName |
| + | NewCases |
| + | TestsDone |
| + | Population |
| | and 9 more... |

O uso de dois modelos pode ser justificado pela necessidade de separar a representação interna dos dados na base de dados da forma como esses dados são apresentados ao utilizador ou consumidor externo.

O modelo **CovidData** pode ser otimizado para atender aos requisitos de armazenamento e manipulação de dados no backend, enquanto o modelo **CovidDataDTO** é projetado para fornecer uma estrutura clara e simplificada para apresentação, evitando o envio de dados excessivos ou sensíveis ao cliente. Essa abordagem favorece a modularidade e a eficiência na comunicação entre os diferentes componentes do sistema.

De seguida foi criado o serviço [*CovidAPI.Services.Rest.CovidDataService*](#) que desempenha um papel importante na aplicação, é responsável por manipular operações relacionadas a dados da COVID-19, incluindo procura, atualização e exclusão de dados.



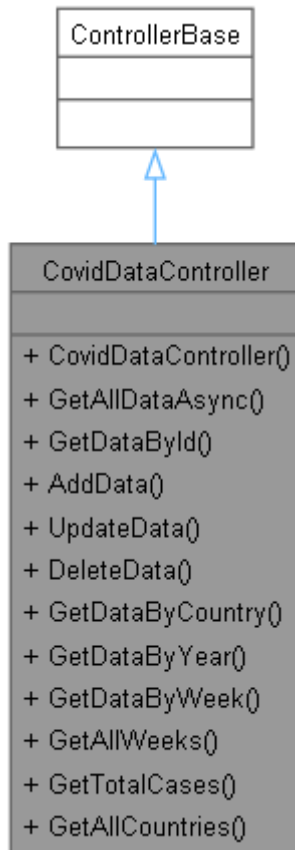
A seta que vai do **CovidDataService** até a interface **ICovidDataService** representa a relação de implementação. Isso significa que o **CovidDataService** está a implementar as funcionalidades definidas na interface **ICovidDataService**. Essa relação é um acordo, onde o serviço promete fornecer as operações especificadas pela interface.

O serviço contém gama abrangente de funções:

| |
|--|
| DataExistsForCountryAndWeekAsync (string country, string week) |
| Checks if data exists for a specific country and week. |
| GetDataByCountryAndWeekAsync (string country, string week) |
| GetAllDataAsync (bool includeGeolocation) |
| Retrieves all COVID-19 data as DTOs, optionally including geolocation information. |
| GetDataByIdAsync (int id, bool includeGeolocation) |
| Retrieves COVID-19 data for a specific ID. |
| GetDataByYearAsync (int year, bool includeGeolocation) |
| Retrieves COVID-19 data for a specific year, optionally including geolocation information. |
| GetDataByWeekAsync (string week, bool includeGeolocation) |
| Retrieves COVID-19 data for a specific week, optionally including geolocation information. |
| AddDataAsync (CovidDataDTO covidDataDTO) |
| Adds COVID-19 data to the database. |
| UpdateDataAsync (CovidDataDTO covidDataDTO) |
| Updates existing COVID-19 data in the database. |
| DeleteDataAsync (int id) |
| Deletes COVID-19 data from the database by ID. |
| GetDataByCountryAsync (string country, bool includeGeolocation) |
| Retrieves COVID-19 data for a specific country, optionally including geolocation information. |
| GetAllWeeksAsync () |
| Retrieves a list of unique weeks available in the CovidData records. |
| GetTotalCasesAsync (bool includeGeolocation) |
| Retrieves total cases data by grouping CovidData records by country and calculating the sum of new cases for each country. |
| GetNewCasesAsync (bool includeGeolocation) |
| Retrieves new cases data by grouping CovidData records by country and week, calculating the sum of new cases for each country and week. |
| GetPopulationDataAsync (bool includeGeolocation) |
| Retrieves population data, including the country and population, with an option to include geolocation information. |
| GetComparisonsAsync (List< string > countries, bool includeGeolocation) |
| Retrieves comparison data for specified countries, with an option to include geolocation information. |
| GetTotalTestsAsync (bool includeGeolocation) |
| Retrieves total tests data, including the country and total tests for the specific year, with an option to include geolocation information. |
| GetGeolocationAsync (bool includeGeolocation) |
| Retrieves geolocation data, including the country and geolocation information, with an option to include geolocation information. |
| GetTestingSourceAsync (bool includeGeolocation) |
| Retrieves testing source data, including the country, testing data source, with an option to include geolocation information. |
| GetTestingRateInfoAsync () |
| Retrieves testing rate information. |
| GetTestsDoneAsync (bool includeGeolocation) |
| Retrieves tests done data, including the country, week, tests done, population, with an option to include geolocation information. |
| GetPositivityRateAsync (bool includeGeolocation) |
| Retrieves positivity rate data, including the country, week, positivity rate, population, with an option to include geolocation information. |
| GetTestingRateAsync (bool includeGeolocation) |
| Retrieves testing rate data, including the country, week, testing rate, population, with an option to include geolocation information. |
| GetAllCountriesAsync () |
| Retrieves a list of all countries. |

O **CovidDataController** é o controlador responsável por operações relacionadas a dados da COVID-19. Atua como uma interface para lidar com requisições relacionadas a esses dados.

O **CovidDataController** é um controlador que herda de **ControllerBase**. No contexto do [ASP.NET Core](#), **ControllerBase** é uma classe base para controladores da Web API.



Funções do controlador:

A seta da imagem que vai do **CovidDataController** para **ControllerBase** indica que **CovidDataController** herda de **ControllerBase**. Essa herança implica que **CovidDataController** possui todas as funcionalidades de **ControllerBase** e, portanto, pode ser tratado como um controlador da Web API no contexto do ASP.NET Core.

Essa relação de herança é fundamental no desenvolvimento do ASP.NET Core, pois fornece ao **CovidDataController** acesso às funcionalidades básicas de um controlador, permitindo que responda a requisições [HTTP](#), manipule parâmetros, retorne resultados e muito mais, de acordo com os padrões da arquitetura [RESTful](#) da Web API.

CovidDataController (*ICovidDataService covidDataService, IGeolocationService geolocationService*)

Initializes a new instance of the **CovidDataController** class.

GetAllDataAsync ()

Gets all COVID-19 data asynchronously.

GetDataById (int id)

AddData ([FromBody] CovidDataDTO covidDataDTO)

Adds new COVID-19 data.

UpdateData (int id, [FromBody] CovidDataDTO covidDataDTO)

Updates COVID-19 data by ID.

DeleteData (int id)

Deletes COVID-19 data by ID.

GetDataByCountry (string country)

Gets COVID-19 data by country.

GetDataByYear (int year)

Gets COVID-19 data by year.

GetDataByWeek (string week)

Gets COVID-19 data by week.

GetAllWeeks ()

Gets all available weeks for COVID-19 data.

GetTotalCases ()

Gets total COVID-19 cases.

GetAllCountries ()

Gets a list of all countries with available COVID-19 data.

Serviços de dados de Geolocalização

É uma componente do sistema responsável por lidar com operações relacionadas à geolocalização. Este serviço é utilizado para interagir com uma API de geolocalização, neste caso a [OpenCage Geocoding API](#) para obter informações sobre determinadas coordenadas geográficas dado a entrada de um nome de um país.

Foi criados múltiplos modelos: [GeolocationApiResponse](#), [GeolocationResult](#), [GeolocationComponents](#) e [Geometry](#).

GeolocationApiResponse representa a resposta da API de geolocalização.

| |
|------------------------|
| GeolocationApiResponse |
| + Results |

```
List< GeolocationResult > Results[get, set]
Gets or sets the list of geolocation results.
```

GeolocationResult representa o resultado de geolocalização.

| |
|-------------------|
| GeolocationResult |
| + Components |
| + Geometry |

```
GeolocationComponents Components[get, set]
Gets or sets the components of the geolocation.

Geometry Geometry[get, set]
Gets or sets the geometry information of the geolocation.
```

GeolocationComponents representa os componentes de uma geolocalização.

| |
|-----------------------|
| GeolocationComponents |
| + Country |

```
string Country[get, set]
Gets or sets the country of the geolocation.
```

Geometry representa as informações de geometria de uma geolocalização.

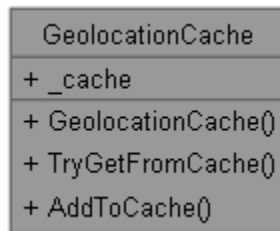
| |
|----------|
| Geometry |
| + Lat |
| + Lng |

```
double Lat[get, set]
Gets or sets the latitude of the geolocation.

double Lng[get, set]
Gets or sets the longitude of the geolocation.
```

O principal propósito desse serviço é facilitar a obtenção, atualização e gestão de dados de geolocalização.

Foi decidido também incluir uma classe [GeolocationCache](#) que representa um cache utilizado para armazenar dados de geolocalização, com o objetivo de evitar chamadas redundantes à API armazenando temporariamente os resultados num ficheiro.



Este diagrama visualiza a estrutura da classe **GeolocationCache** e como ela lida com a adição e consulta de dados de geolocalização em um cache interno.

```

bool TryGetFromCache (string country, out GeolocationApiResponse geolocationResponse)
    Tries to retrieve geolocation data from the cache for a specific country.

void AddToCache (string country, GeolocationApiResponse geolocationResponse)
    Adds geolocation data to the cache for a specific country.
  
```

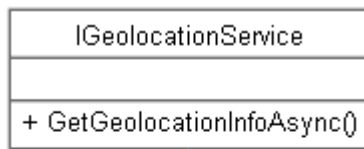
Aqui está um exemplo de como os dados são armazenados no ficheiro **cache.json** para a Chéquia:

```

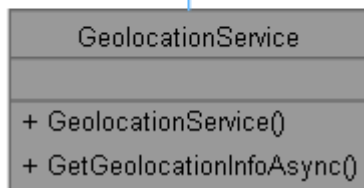
{"Czechia":{"Results":[{"Components":{"Country":"Czechia"},"Geometry":{"Lat":49.7439047,"Lng":15.3381061}}]}
  
```

O pedido completo de dados covid-19 junto com a integração do serviço de geolocalização vai ser futuramente exemplificado.

A classe [GeolocationService](#) representa o serviço para obter informações de geolocalização específicas de um país



A seta no gráfico, de **GeolocationService** para **IGeolocationService**, indica que a classe **GeolocationService** implementa a interface **IGeolocationService**. Isto promove flexibilidade e manutenção de código.



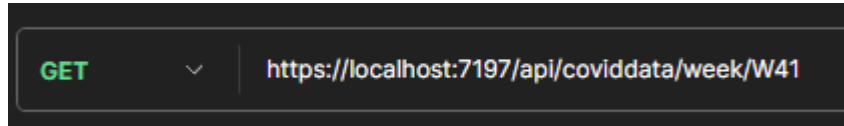
```

GetGeolocationInfoAsync (string country)
    Retrieves geolocation information asynchronously for a specific country.
  
```

Em suma, a integração do serviço de geolocalização com o serviço de dados COVID cria uma sinergia valiosa para o sistema. Ao unir esses serviços, podem-se enriquecer os dados epidemiológicos com informações geográficas relevantes. Isso não apenas aprimora a precisão das análises e visualizações, mas também oferece insights geolocalizados cruciais para entender a propagação da COVID-19.

Demonstração dos serviços dados Covid e Geolocalização.

Utilizando o [Postman](#), aqui está a exemplificação de um Pedido GET de dados da semana 41:



```
{
  "id": 4,
  "country": "Czechia",
  "countryCode": "CZ",
  "year": 2022,
  "week": "W41",
  "region": "CZ",
  "regionName": "Czechia",
  "newCases": 19787,
  "testsDone": 62029,
  "population": 10516707,
  "positivityRate": 31.899595350561835,
  "testingRate": 589.8139027739386,
  "testingDataSource": "TESSy COVID-19",
  "totalCasesYear": 197870,
  "totalTestsYear": 620290,
  "perCapitaCases": 0.01881482483062426,
  "perCapitaTests": 0.058981390277393866,
  "geolocation": {
    "country": "Czechia"
  },
  "geometry": {
    "lat": 49.7439047,
    "lng": 15.3381061
  }
},
```

```
{
  "id": 55,
  "country": "Croatia",
  "countryCode": "HR",
  "year": 2022,
  "week": "W41",
  "region": "HR",
  "regionName": "Croatia",
  "newCases": 6664,
  "testsDone": 37846,
  "population": 3862305,
  "positivityRate": 17.60820165935634,
  "testingRate": 979.8811849400811,
  "testingDataSource": "TESSy COVID-19",
  "totalCasesYear": 66640,
  "totalTestsYear": 378460,
  "perCapitaCases": 0.01725394550663399,
  "perCapitaTests": 0.09798811849400811,
  "geolocation": {
    "country": "Croatia"
  },
  "geometry": {
    "lat": 45.3658443,
    "lng": 15.6575209
  }
},
```

Esta resposta é um conjunto de dados formatados em JSON, representando informações específicas sobre a situação da COVID-19.

Contém um identificador único para o conjunto de dados, o nome do país associado aos dados, o código do país, o ano ao qual os dados se referem, a semana dentro do ano, o nome e código da região (não explorada neste projeto mas pode ser utilizado para melhor precisão de dados), novos casos da semana, numero total de testes da semana, população total do país, taxa de positividade, taxa de testagem (número de testes realizados por 1000 pessoas), fonte de dados, número total de casos registados durante o ano, número total de testes realizados durante o ano, número de casos e testes por pessoa, e informações de geolocalização/coordenadas geográficas do país (latitude e longitude).

Serviços de Autenticação

Para o serviço de autenticação foram definidos três modelos: [CovidAPI.Models.User](#), [CovidAPI.Models.RegisterRequestDTO](#) e [CovidAPI.Models.LoginRequestDTO](#).

User armazena informações sobre os utilizadores.

| CovidAPI.Models.User | |
|----------------------|--|
| + Id | int Id [get, set] Gets or sets the unique identifier for the user. |
| + Username | string Username [get, set] Gets or sets the username of the user. |
| + PasswordHash | byte[] PasswordHash [get, set] Gets or sets the hash of the user's password. |
| + PasswordSalt | string PasswordSalt [get, set] Gets or sets the salt used for password hashing. |

RegisterRequestDTO é utilizado para transferir dados relacionados ao registo de um

| CovidAPI.Models.RegisterRequestDTO | |
|------------------------------------|---|
| + Username | string Username [get, set] Gets or sets the username for the registration request. |
| + Password | string Password [get, set] Gets or sets the password for the registration request. |

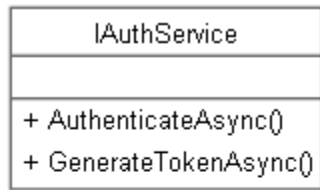
utilizador.

LoginRequestDTO também é utilizado para transferir dados, mas desta vez para solicitações de login.

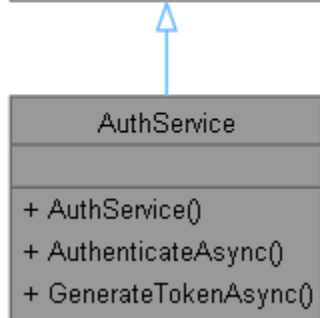
| CovidAPI.Models.LoginRequestDTO | |
|---------------------------------|--|
| + Username | string Username [get, set] Gets or sets the username for the login request. |
| + Password | string Password [get, set] Gets or sets the password for the login request. |

A junção dos três modelos cria uma estrutura coesa para o serviço de autenticação na aplicação.

A classe [AuthService](#) é responsável pela autenticação do utilizador e pela criação de tokens [JWT](#).



A seta indica que a classe **AuthService** implementa todos os membros (métodos, propriedades, etc.) definidos pela interface **AuthService**.



AuthenticateAsync (string username, string password)

Authenticates a user based on the provided username and password.

GenerateTokenAsync (User user)

Generates a JWT token for the specified user.

Esta classe “trabalha” com outros serviços para garantir a segurança e eficiência de autenticação da aplicação:

Com a classe [PasswordService](#), que é responsável por operações relacionadas à password, como o hash da password, verificação de password e criação de salt utilizando [BCrypt](#) (biblioteca de encriptação).

A seta indica que a classe **PasswordService** implementa todos os membros (métodos, propriedades, etc.) definidos pela interface **IPasswordService**.



byte[] HashPassword (string password, out string salt)

Hashes the provided password and generates a salt using BCrypt.

bool VerifyPassword (string password, byte[] storedHash, string storedSalt)

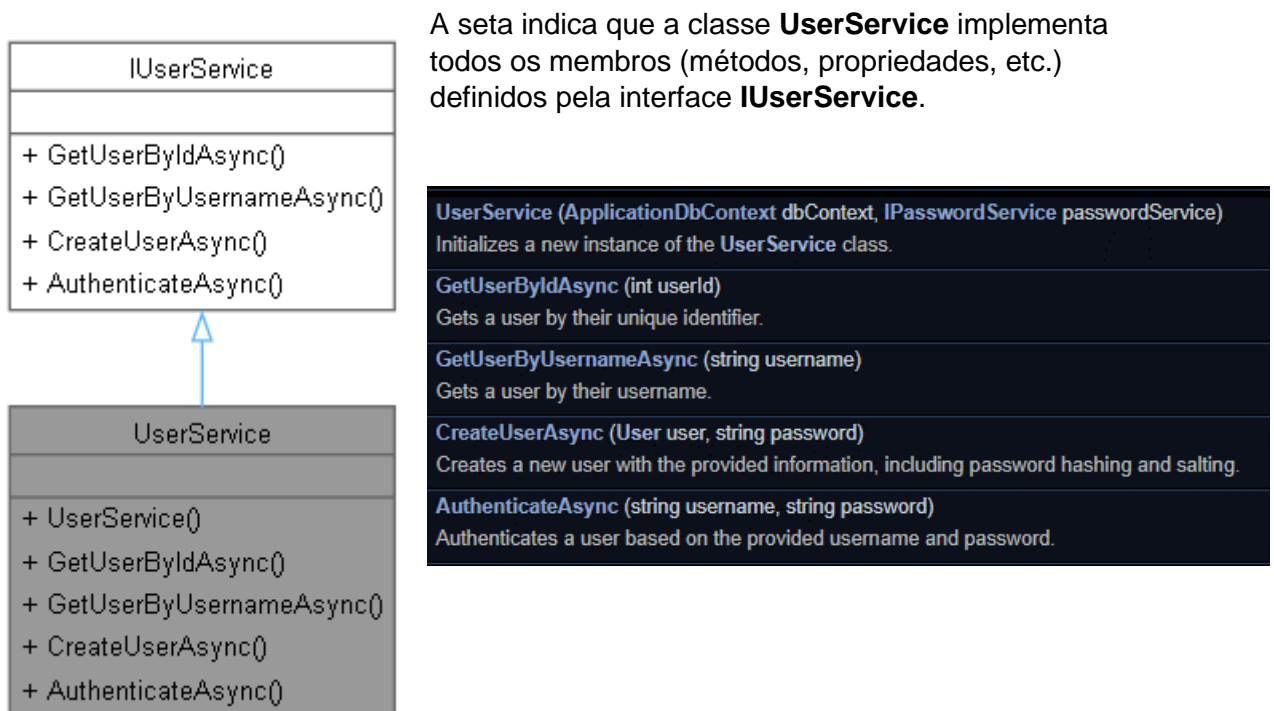
Verifies the provided password against the stored hash and salt using BCrypt.

string GenerateSalt ()

Generates a new salt using BCrypt.

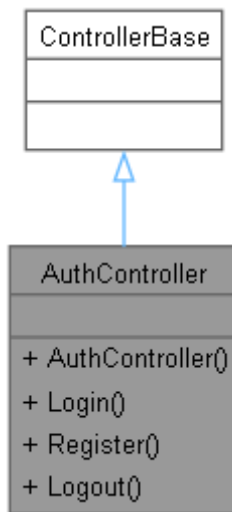
Estas operações são úteis em contextos de autenticação, onde é crucial armazenar passwords de forma segura e eficiente, minimizando riscos de exposição.

A classe **UserService** é responsável por operações relacionadas aos utilizadores incluindo autenticação e recuperação de informações do utilizador.



Estas operações são essenciais para a gestão de utilizadores, permitindo a autenticação segura, recuperação de informações do utilizador e criação de novos utilizadores no sistema. O uso de operações assíncronas reflete a natureza potencialmente demorada de algumas dessas operações, como a interação com uma base de dados.

A classe [AuthController](#) é o controlador responsável por operações relacionadas à autenticação.



O **AuthController** é um controlador que herda de **ControllerBase**, já explicado anteriormente

A seta da imagem que vai do **AuthController** para **ControllerBase** indica que **AuthController** herda de **ControllerBase**. Essa herança implica que **AuthController** possui todas as funcionalidades de **ControllerBase** e, portanto, pode ser tratado como um controlador da Web API no contexto do ASP.NET Core.

```
Login ([FromBody] LoginRequestDTO request)
Logs in a user.

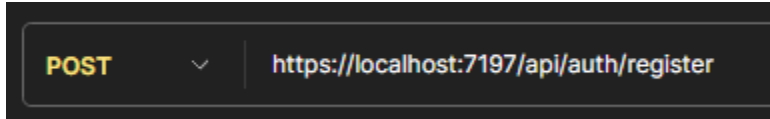
Register ([FromBody] RegisterRequestDTO request)
Registers a new user.

Logout ()
Logs out the currently authenticated user.
```

Estas operações são fundamentais para a gestão da autenticação de utilizadores. O método `Login` gera um token JWT após autenticar com sucesso um utilizador. O método `Logout` encerra a sessão do utilizador autenticado. O método `Register` cria um novo utilizador e fornece um token JWT em caso de sucesso.

Demonstração serviços de Autenticação

Novamente, utilizando o [Postman](#), aqui está a exemplificação do registo e login de um utilizador.

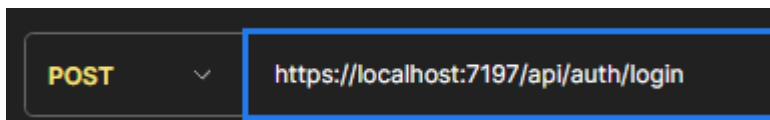


Entrada:

```
{
  "username": "exemplo",
  "password": "exemplo"
}
```

Resposta: Atribuição de um ID e mensagem de sucesso

```
{
  "message": "Registration successful",
  "userId": 19,
}
```



Resposta: Token JWT é criado

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bm1xdWVfbmFtZSI6ImV4ZW1wbG8iLCJ0eXciOiE3MDM5NTAyNDMsImV4cCI6MTcwMzk1Mzg0MywiaWF0IjoxNzAzOTUwMjQzLCJpc3MiOiJ5b3VyX2lzc3Vlc19oZXJlIiwiaXVkiOiJoieW91c19hdWRpZW5jZV9oZXJlIn0.mLiz2i2NqcA5jxRCZxdx5Z74C9MfKSPd8Tz23lEtAd4"
}
```

após login.

O utilizador é registado na base de dados com uma password encriptada.

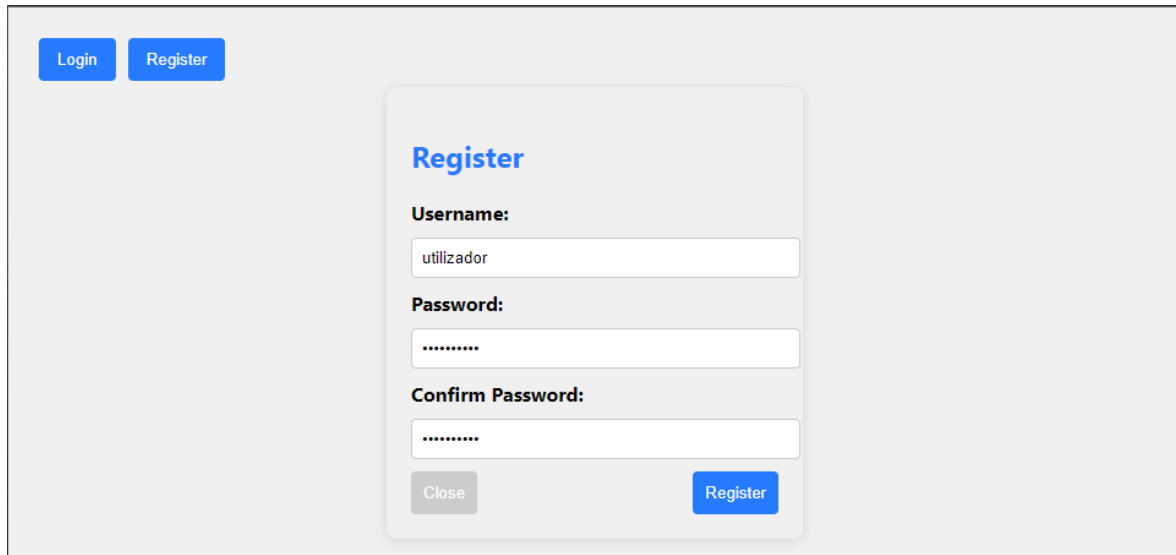
| | | | |
|------|---------|------|----------------------------------|
| 18 | exemplo | BLOB | \$2a\$11\$4humxxPFxvhIs1lh7/ISf. |
| 19 | exemplo | BLOB | \$2a\$11\$4humxxPFxvhIs1lh7/ISf. |
| NULL | NULL | NULL | NULL |

Dashboard

Para a criação da Dashboard foi utilizada a framework [React](#) em javascript.

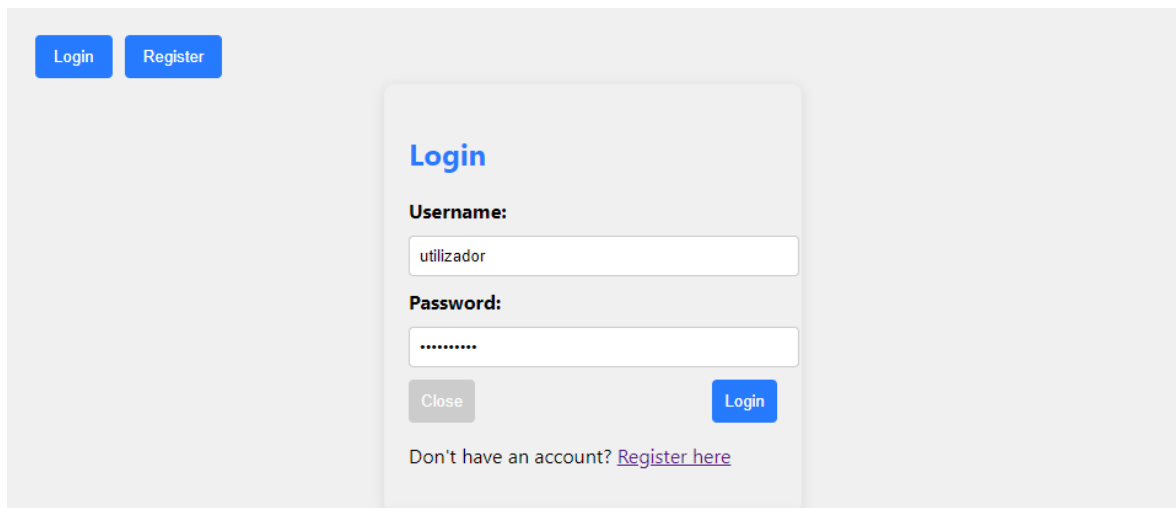
Foi utilizada a biblioteca [axios](#) para interagir com os endpoints da API, a biblioteca [leaflet.js](#) para a utilização de mapas em conjunto com o [openstreetmap](#), e a biblioteca [recharts](#) para inclusão de gráficos.

Login/Registo:



The image shows a web interface for a registration form. At the top left, there are two blue buttons: "Login" and "Register". The main content is a light gray box containing a "Register" form. The form has a title "Register" in blue. It includes three input fields: "Username:" with the value "utilizador", "Password:" with masked characters "*****", and "Confirm Password:" with masked characters "*****". At the bottom of the form, there are two buttons: a gray "Close" button and a blue "Register" button.

20 utilizador BLOB \$2a\$11\$VhdO9G/BcB7XE1xvZ0mou

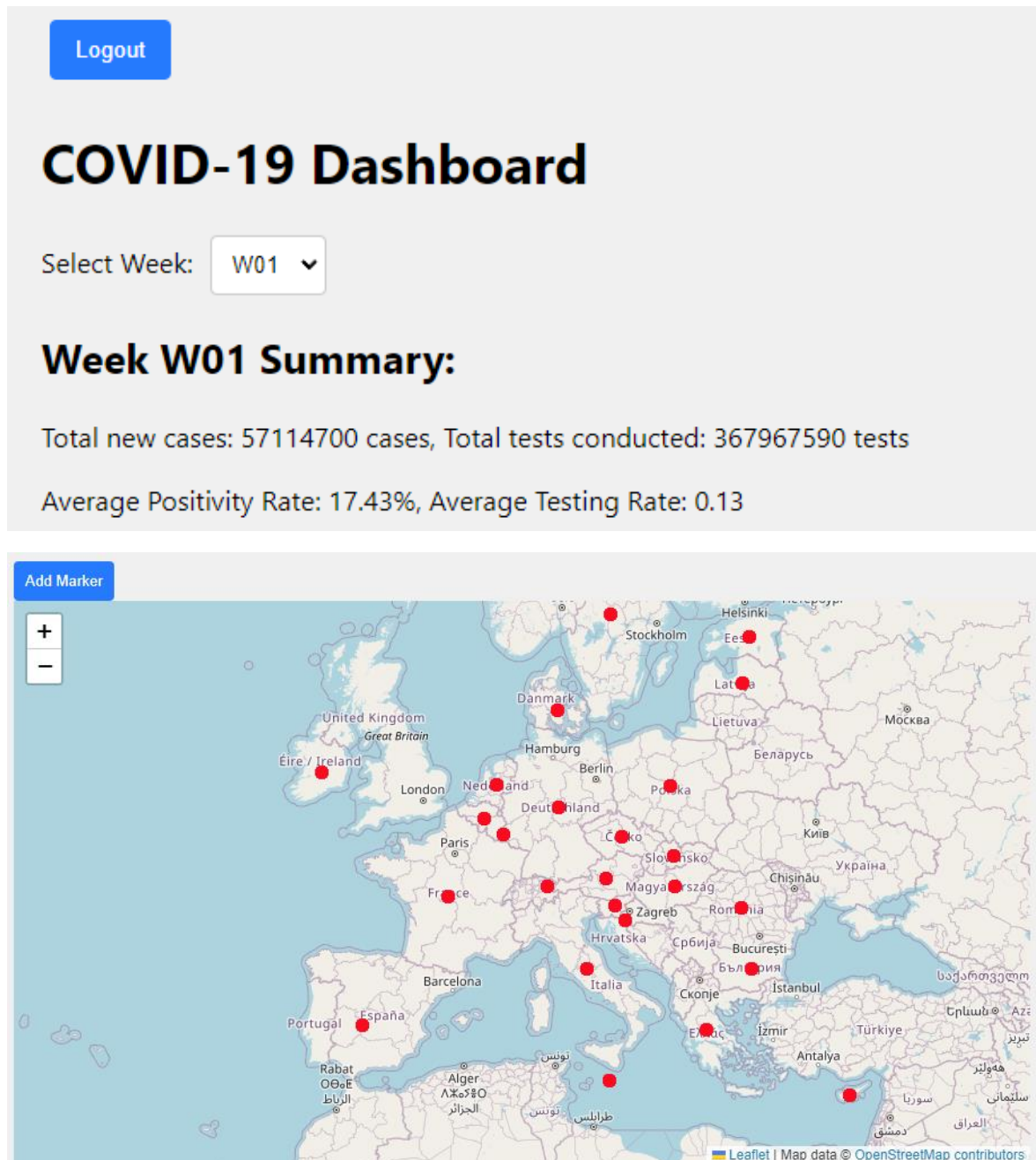


The image shows a web interface for a login form. At the top left, there are two blue buttons: "Login" and "Register". The main content is a light gray box containing a "Login" form. The form has a title "Login" in blue. It includes two input fields: "Username:" with the value "utilizador" and "Password:" with masked characters "*****". At the bottom of the form, there are two buttons: a gray "Close" button and a blue "Login" button. Below the form, there is a link: "Don't have an account? [Register here](#)".

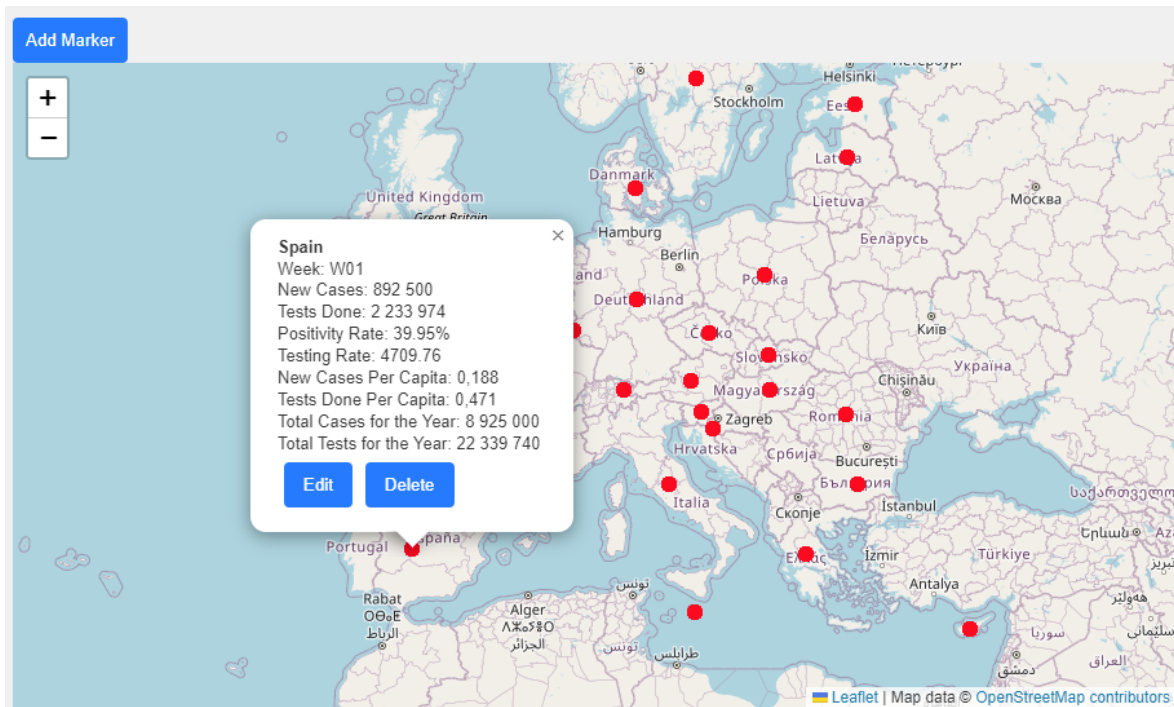
Página Principal:

Na página principal é possível selecionar uma semana e ver os dados de cada país num mapa e algumas métricas referentes aos dados covid dessa semana. Cada país é marcado com um ponto na devida latitude e longitude fornecida pela API.

Por exemplo, semana 1



Ao clicar num ponto, dados sobre a semana nesse país, são mostrados.



É possível adicionar pontos também, por exemplo, Portugal não tem um ponto na semana 1, se criarmos um ponto com dados na semana 1 para Portugal a API automaticamente vai buscar os dados relativos à geolocalização em Portugal, guarda na cache da API e o ponto é assim colocado com os dados relativos.

Foi gerado o ponto para Portugal na semana 1:

Add Marker

Country:
Portugal

Country Code:
PT

Year:
2022

Week:
W01

Region:
Portugal

Region Name:
PT

New Cases:
1000

Tests Done:
100

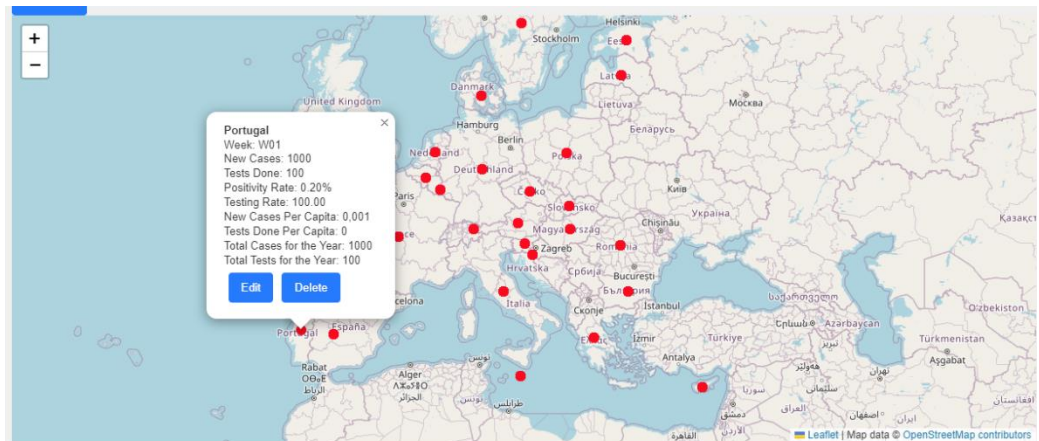
Population:
999999

Positivity Rate:
0.2

Testing Rate:
100

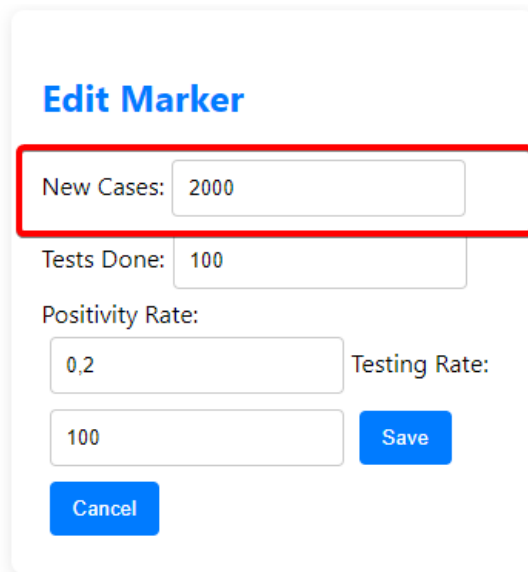
Testing Data Source:
Tessy

Cancel **Add Marker**



Também é possível apagar e atualizar pontos:

Atualização do valor de novos casos em Portugal:



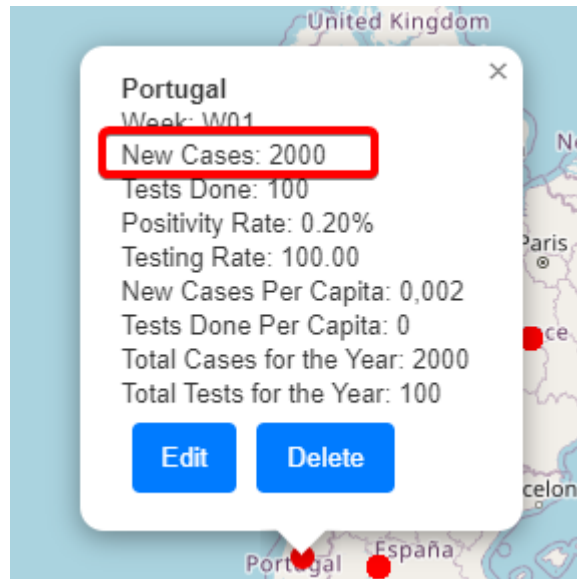
Edit Marker

New Cases:

Tests Done:

Positivity Rate:

Testing Rate:



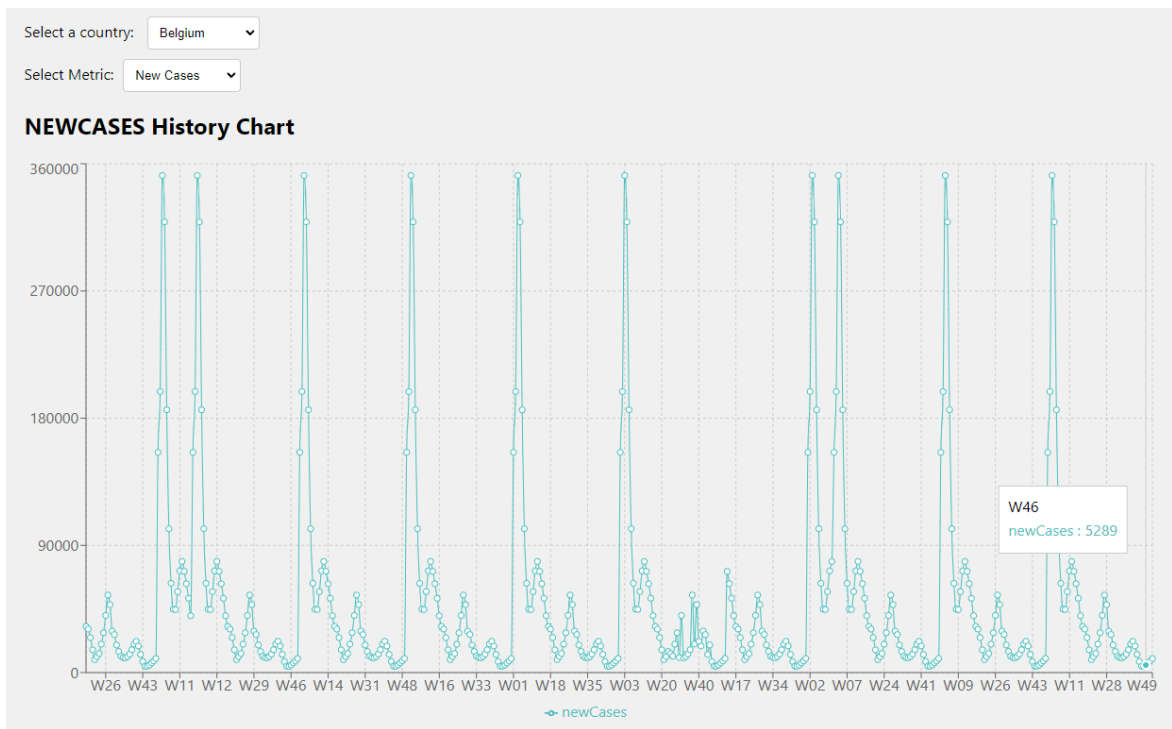
Deleção do ponto:



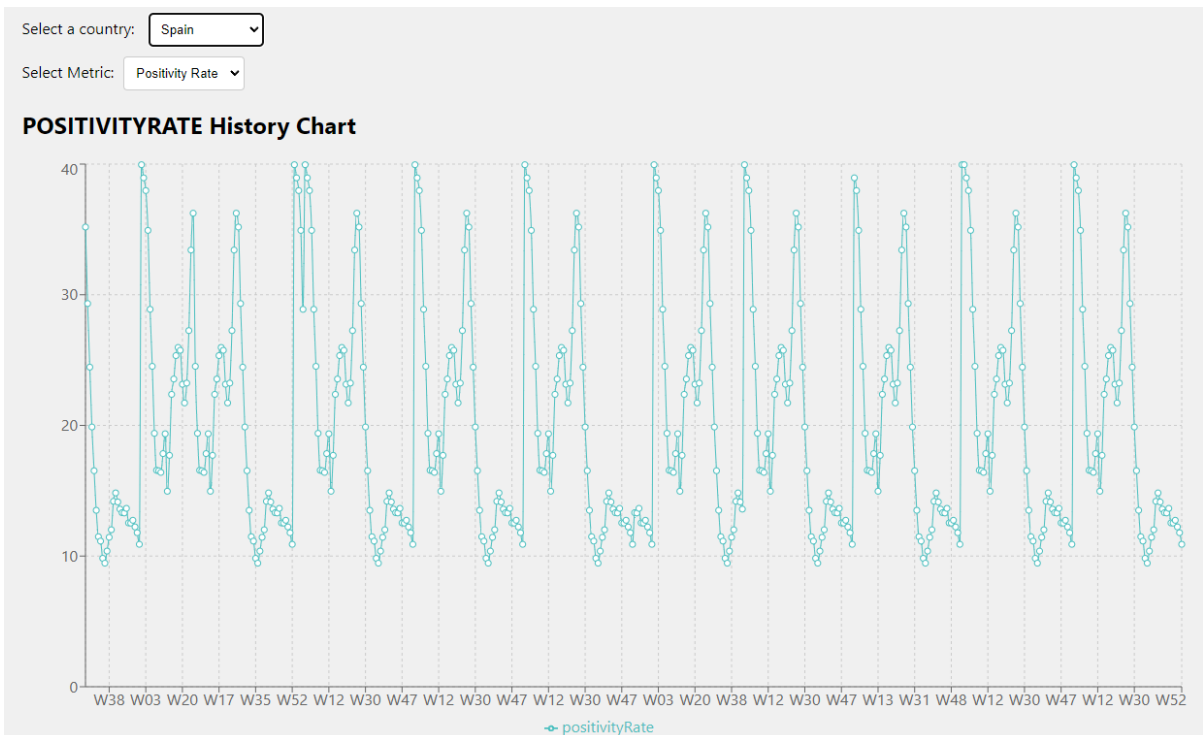
Isto coloca em prática uma variedade de rotas e funções presentes na API .

Mais abaixo, também se pode selecionar um país e a respetiva métrica para listar os dados relativos a todas as semanas desse ano num gráfico.

Por exemplo, Bélgica – Novos Casos

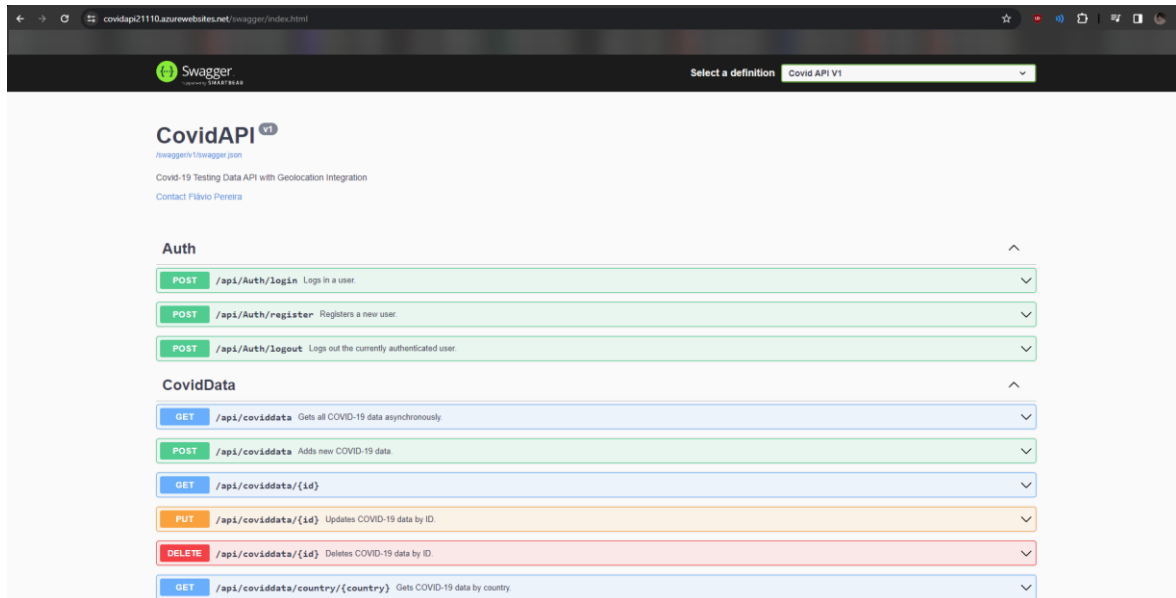


Apesar de o gráfico não estar a 100% e estar praticamente a mostrar dados em loop, demonstra que é possível utilizar os dados para obter informação histórica. Outro exemplo:



Publicação na Cloud

A publicação no [Microsoft Azure](#) foi um pouco problemática, umas vezes publicava outras não, era mostrado um erro de swagger e um pacote .net que ficou por resolver, mas apesar dos endpoints não funcionarem na Cloud, o swagger foi lido.



No geral gostava bastante de experimentar trabalhar em Cloud visto que é algo que me despertou interesse mas infelizmente não foi possível explorar tudo adequadamente neste projeto.

Também procurei utilizar [Docker](#) mas não tive essa oportunidade.

Conclusão

O desenvolvimento deste projeto abordou de forma abrangente a gestão e análise de dados relacionados à pandemia de Covid-19, enfatizando a integração de diversas ferramentas e serviços. A seguir, destacam-se algumas conclusões relevantes:

- **Abordagem Abrangente:** O projeto envolveu uma extensa coleta de dados da pandemia de Covid-19 na Europa, com foco no ano de 2022. A utilização de um serviço de geolocalização complementou esses dados, permitindo uma representação visual em mapas.
- **Integração de Sistemas e Ferramentas:** A Fase 1 do projeto demonstrou a utilidade das ferramentas ETL (Extract, Transform, Load) e solidificou os conceitos de integração de sistemas. A integração de serviços web, APIs externas e a apresentação por meio de um painel contribuíram para uma compreensão mais ampla.
- **Documentação Eficiente:** O processo de documentação, realizado pelo Doxygen, proporcionou uma visão clara da estrutura do código-fonte. A documentação em HTML e PDF facilita a compreensão e manutenção do projeto.
- **Serviços de Dados Covid e Geolocalização:** A implementação de serviços dedicados aos dados da Covid-19 e geolocalização trouxe uma abordagem modular. A separação de modelos internos e externos permitiu uma melhor adaptação às necessidades de armazenamento e apresentação.
- **Demonstrações Práticas:** A utilização do Postman para exemplificar solicitações GET de dados e autenticação ofereceu uma visão prática da funcionalidade da API. As demonstrações na Dashboard, com interação de mapas e gráficos, adicionaram uma camada visual e interativa aos dados.
- **Serviços de Autenticação:** A implementação de um serviço de autenticação, com modelos específicos e criação de tokens JWT, contribuiu para a segurança e controle de acesso à aplicação.
- **Desafios e Oportunidades Futuras:** Os desafios encontrados na publicação na nuvem e a exploração limitada do Docker representam oportunidades de aprimoramento futuro. A utilização de serviços em nuvem, como o Microsoft Azure, pode ser uma direção promissora para a expansão do projeto.

Em suma, o projeto não apenas atingiu seus objetivos principais relacionados à análise de dados da Covid-19, mas também evidenciou a importância da integração de serviços, documentação eficiente e a aplicação prática de conceitos em um contexto mais amplo.