Wymagania funkcjonalne

Strona pobierające dane osób szukających pracy w sektorze IT

1. Zespół realizujący projekt:

Imię i nazwisko	Rola	Kontakt
Oskar Aleksanderek	Kierownik zespołu	o.aleksanderek.615@studms.ug.edu.pl
Adrian Jarząb	Deweloper	a.jarzab.536@studms.ug.edu.pl

2. Wprowadzenie:

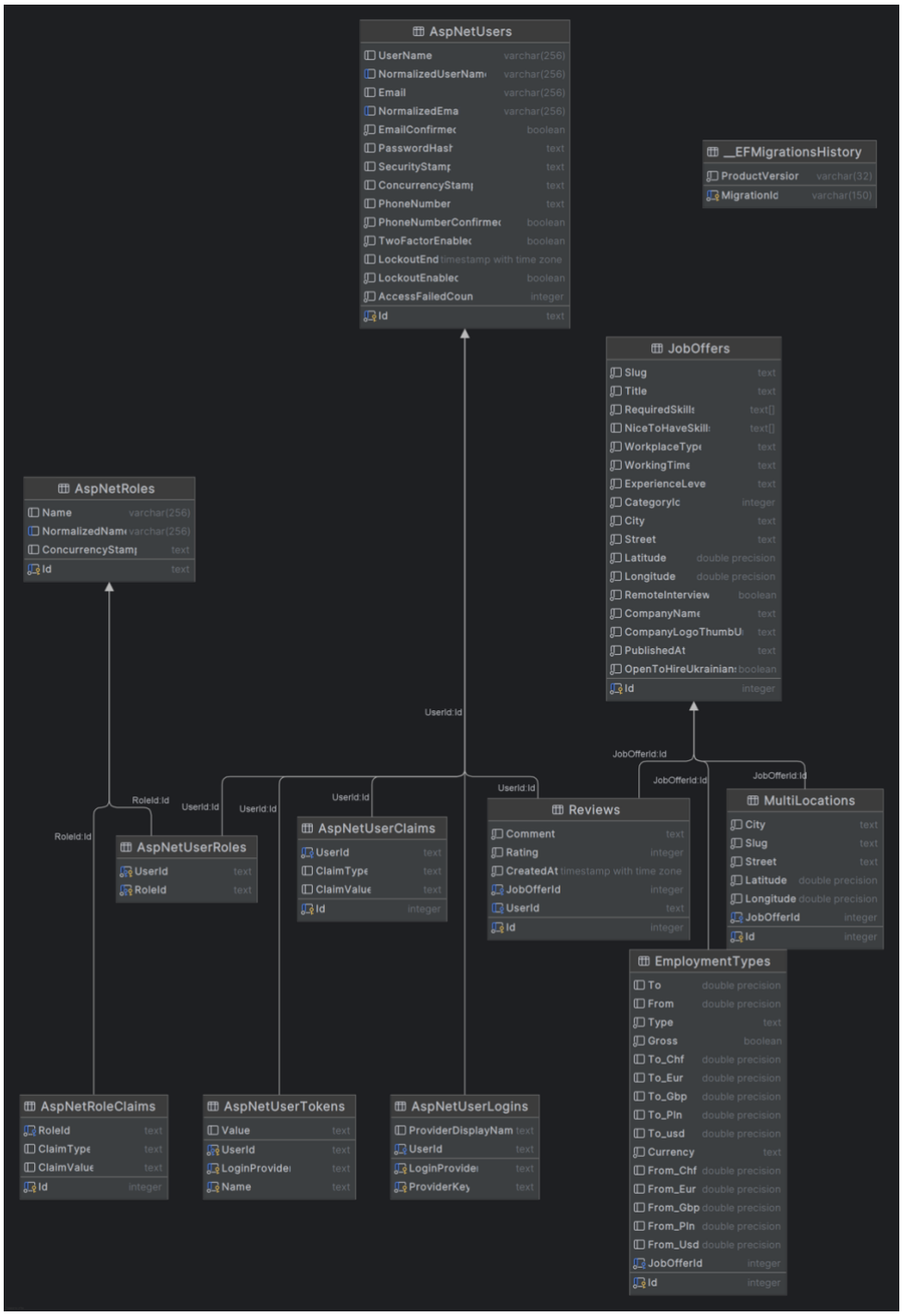
Dużo osób w dzisiejszych czasach zmaga się ze znalezieniem dla siebie idealnej pracy. Przychodzimy z rozwiązaniem tego problemu. Projekt będzie proof of concept większej strony internetowej, która poprzez osobne serwisy będzie zbierała dane z różnych stron internetowych na temat ofert prac w IT. Użytkownicy będą mogli podzielić się swoją opinią na temat danej oferty, w tym również firmy. Zdecydowanie przyniesie to korzyść osobą poszukującym pracę, które będą mogły wykształcić na pierwszy rzut oka opinię o danej firmie oraz ich ofertach.

3. Szczegółowe wymagania funkcjonalne systemu:
   - **3.1.** Serwisy scrapujące oferty pracy ze stron posiadającymi oferty pracy w IT
   - **3.2.** Baza danych przechowująca dane ze scrapera
   - **3.3.** Walidacja danych przez stronę backendową
   - **3.4.** Wyświetlenie wszystkich ofert w głównym widoku
   - **3.5.** Zakładka opisowa stronę
   - **3.6.** Zakładka z kontaktem
   - **3.7.** Widok szczegółów danej oferty
   - **3.8.** Wyświetlenie opinii
   - **3.9.** Dodawanie opinii
   - **3.10.** Usuwanie opinii
   - **3.11.** Edycja opinii
   - **3.12.** Walidacja operacji związanych z opiniami
   - **3.13.** Dane mają być serwowane dla frontend'u poprzez interfejs programistyczny
   - **3.14.** Komunikacja front-back poprzez interfejs programistyczny

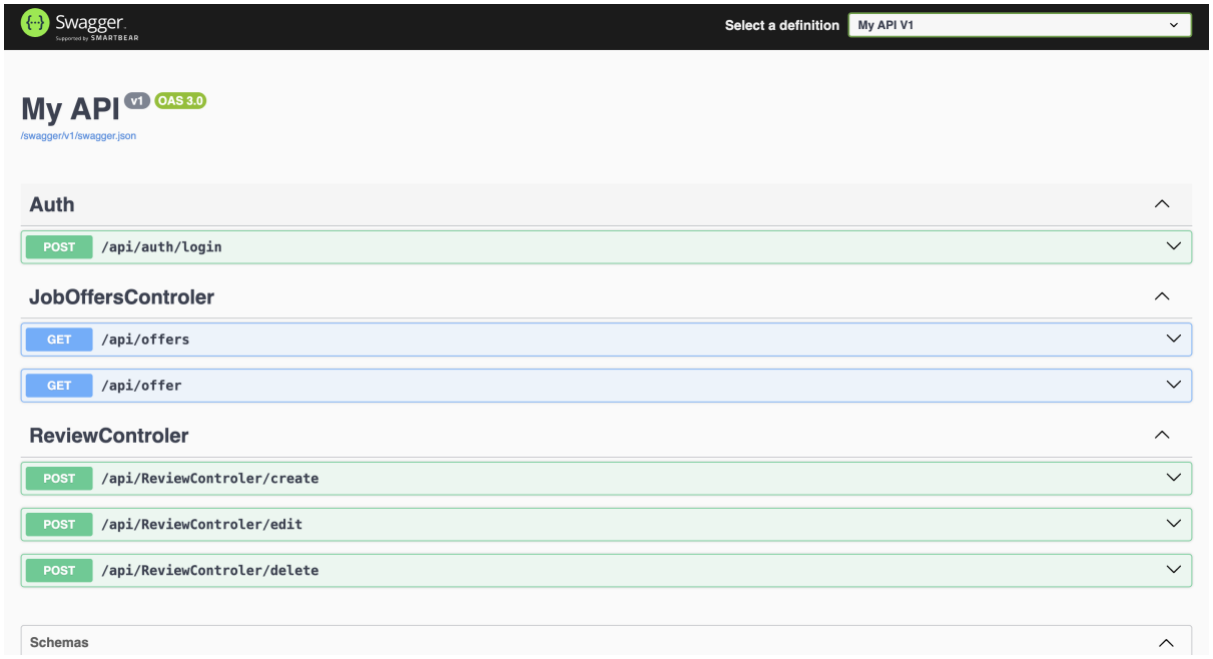
4. Techonolgie
   - 4.1. Frontend
     - 4.1.1. React
     - 4.1.2. Typescript
   - 4.2. Backend

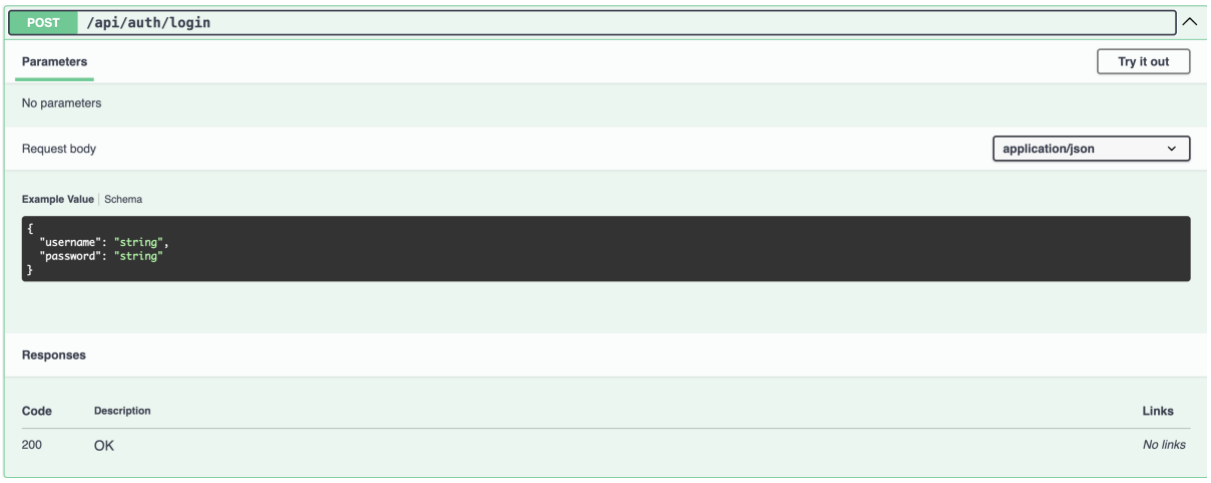
4.2.1. ASP.NET
4.2.2. C#/.NET
4.2.3. PostgreSQL

5. Diagram ERD



6. Dokumentacja API (Swagger)


Swagger
Supported by SMARTBEAR
Select a definition
My API V1
My API
V1
OAS 3.0
/swagger/v1/swagger.json
Auth
POST /api/auth/login
JobOffersControler
GET /api/offers
GET /api/offer
ReviewControler
POST /api/ReviewControler/create
POST /api/ReviewControler/edit
POST /api/ReviewControler/delete
Schemas


POST /api/auth/login
Parameters
Try it out
No parameters
Request body
application/json
Example Value | Schema
{
"username": "string",
"password": "string"
}
Responses
Code
Description
Links
200
OK
No links


GET /api/offers
Parameters
Try it out
Name
Description
page
integer($int32)
(query)
page
Responses
Code
Description
Links
200
OK
No links


GET
/api/offer
Parameters
Try it out
Name
Description
id
integer($int32)
(query)
id
Responses
Code
Description
Links
200
OK
No links


POST
/api/ReviewControler/create
Parameters
Try it out
No parameters
Request body
application/json
Example Value
Schema
{
"jobOfferId": 0,
"comment": "string",
"rating": 0
}
Responses
Code
Description
Links
200
OK
No links


POST
/api/ReviewControler/edit
Parameters
Try it out
No parameters
Request body
application/json
Example Value
Schema
{
"reviewId": 0,
"comment": "string",
"rating": 0
}
Responses
Code
Description
Links
200
OK
No links


POST
/api/ReviewControler/delete
Parameters
Try it out
No parameters
Request body
application/json
Example Value | Schema
{
"reviewId": 0
}
Responses
Code
Description
Links
200
OK
No links