Bydgoszcz 12.09.18.

Stworzenie platformy do oddawania i sprawdzania sprawozdań.

Wydział Telekomunikacji, Informatyki i Elektrotechniki

Kierunek Informatyka Stosowana grupa 7

Narzędzia programistyczne

prowadzący Vasyl Zaiats

Mateusz Tokarczyk

Wiktor Kawałek

Jakub Olszak

Antoni Malak

https://github.com/mati1000500100900/Sprawozdania-API

Temat projektu:Stworzenie platformy do oddawania i sprawdzania sprawozdań.

1.Cele:

Celem projektu jest stworzyć platformę ułatwiającą pracę ze sprawozdaniami wykładowcom i studentom. Strona studenta i wykładowcy będą się różnić funkcjami oraz będą dostosowane do ich potrzeb.

2.Funkcje:

Wszyscy użytkownicy będą musieli posiadać swoje prywatne konta, będą one miały inne role zależnie czy to konto studenta, czy wykładowcy.

Studenci będą mogli w prosty sposób przesyłać sprawozdania. Obsługiwany będzie system kursów, do których można będzie się zapisać, dzięki czemu będzie można sprawdzać jakie sprawozdania mamy do zrobienia, sprawdzić oceny jakie dostaliśmy za poprzednie które już oddaliśmy.

Wykładowcy będą mogli tworzyć, prowadzić kursy, dodawać sprawozdania z instrukcjami do zrobienia przez studentów. Strona będzie ułatwiać im przeglądanie oraz ocenę sprawozdań.

3.Użyte technologie

Do nadania odpowiedniego wyglądu strony użyliśmy bootstrapa, dzięki któremu w szybki sposób można stworzyć czytelną i przejrzysty wygląd. Jako platformę do budowy aplikacji strony wybraliśmy Angular w wersji 5. Przeznaczony jest do tworzenia SPA, dzięki czemu dociąganie danych będzie prostsze i nie będzie wymagało przeładowywania strony. Jest łatwy do opanowania, dzięki wielu oficjalnym poradnikom. Naszą bazą danych będzie mySQL, ponieważ jest uniwersalnym, znanym przez wszystkich rozwiązaniem. Do jej obsługi nie mogliśmy używać języka PHP, co zmusiło nas do wymyślenia innego rozwiązania. Za pomocą REST api, Angular wysyła odpowiednie zapytania json których nasłuchuje Spring, oraz który już po stronie serwera wykonuje operacje na bazie danych.

4.Działanie projektu:

4.1.Frontend:

Wszystkie akcje na stronie są obsługiwane najpierw przez Angulara. Przy logowaniu poprzez REST api wysyła zapytanie w formacie json w którym są dane logowania opisane poprzez odpowiednią klasę zawierającą email oraz hasło użytkownika. Zapytanie składa się z trzech części. Pierwsza, header zawiera informacje w jaki sposób są szyfrowane dane. Payload zawiera przesyłane zaszyfrowane dane. Signature służy walidacji tokena. Hasła są szyfrowane za pomocą gotowej funkcji przez angulara jeszcze przed wysłaniem.

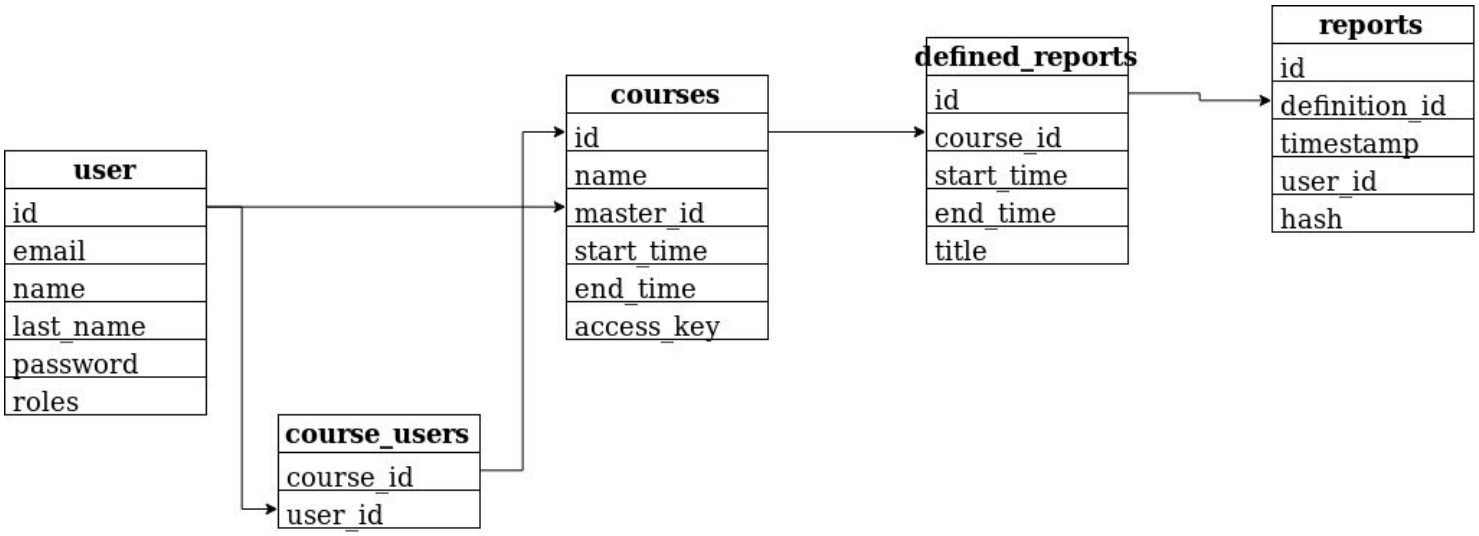
4.2.Backend:

Za nasłuchiwanie zapytań odpowiedzialny jest Spring, który zależnie od tego czy zapytanie jest poprawne czy nie, wysyła adekwatną odpowiedź zwrotną. Wszystko jest obsługiwane przez tylko jeden endpoint, a wszystkie zapytania są podstawowymi metodami przekazywania danych takimi jak GET, POST, oraz PATCH, UPDATE W przypadku pomyślnej rejestracji odpowiedzią jest wiadomość “registered”. Gdy się zalogujemy zwracany jest token JWT który zawiera informacje na temat id użytkownika, który jest

zapisywany w session storage przeglądarki. Potem dzięki temu tokenowi można wysłać zapytanie o informacje zalogowanego użytkownika, oraz na jego podstawie zmienia się układ strony. Niektóre opcje strony są widoczne tylko dla studentów a niektóre tylko dla wykładowców. Tokeny są sprawdzane tylko przy logowaniu. Dodatkowo niektóre zapytania są realizowane jedynie dla użytkowników o odpowiedniej roli. Na przykład tylko wykładowcy mogą tworzyć kursy. Reszta komend działa na podobnej zasadzie, są one wylistowane w dokumentacji.

4.3.Przechowywanie danych.

Do przechowywania danych używamy bazy mySql, wykonujemy na niej operacje z pomocą Springa który wykonuje odpowiednie akcje zależnie od zapytania, dodaje dane, usuwa je, albo tylko je sprawdza. W bazie danych są przechowywane dane logowania użytkowników, ich adresy email oraz zahashowane hasła. Przechowywujemy również informacje o kursach, i ich uczestnikach. Schemat bazy danych wygląda następująco.



Sprawozdania API

Logowanie i Rejestracja

POST /user/register
podajemy obiekt json

```
{
   "email": "JanKow001@utp.edu.pl",
   "name": "Jan",
   "last_name": "Kowalski",
   "password": "123456"
}
```

zwraca JSON "registered"

POST /user/login
podajemy obiekt json

```
{
   "email": "AdaNow002@utp.edu.pl",
   "password": "r00t"
}
```

zwraca JSON token JWT

GET /user/whoami
podajemy, tylko token JWT w headrze

Authorization: Bearer <token>

zwraca JSON z danymi zalogowanego usera

Kursy

GET /courses
nie podajemy nic

zwaca JSON ze wszystkimi kursami

GET /courses/{id}
podajemy {id} w URI

zwaca JSON z jednym kursem

POST /courses
podajemy objekt JSON

```
{
   "name": "Narzędzia programistyczne",
   "start_time": "1970-01-01 00:00",
   "end_time": "1970-01-01 23:59"
```

}

zwraca JSON "Added new course"

PATCH /courses/{id}
podajemy id w URI i JSON ze zmianami

```
{
   "name": "Nowa nazwa"
}
```

zwraca JSON "updated'

DELETE /courses/{id}
podajemy w URI id kursu do usnięcia

zwraca JSON "deleted"

POST /courses/join/{access_key}
podajemy{access_key} kursu do którego chcemy dołączyć, zwraca JSON "joined"

GET /courses/my
nic nie podajemy

zwraca JSON z kursami zalogowanego usera

Definicje sprawozdań

GET /courses/definitions
nie podajemy nic

zwraca JSON ze wszystkimi definicjami

POST /courses/definitions
podajemy JSON z nową definicją, możemy dodawać tylko z rolą TEACHER, tylko do swoich kursów

```
{
   "course_id": 2,
   "title": "Obsługa GIT-a",
   "start_time": "1970-01-01 00:00",
   "end_time": "1970-01-01 23:59"
}
```

zwraca JSON "added"

GET /courses/{id}/definitions
podajemy w URI {id} kursu

zwraca JSON ze wszystkimi definicjami tego kursu

GET /courses/definitions/{id}

podajemy w URI {id} definicji

zwraca JSON z tą definicją

PATCH /courses/definitions/{id}
podajemy JSON z tym co chcemy zmienić i {id} definicji w URI, jak przy dodawaniu można edytować tylko swoje definicje

```
{
   "start_time": "1970-01-01 04:20",
}
```

zwraca JSON "updated"

DELETE /courses/definitions/{id}
podajemy {id} definicji w URI, można usuwać tylko swoje definicje

zwraca JSON "deleted"

GET /courses/definitions/my
podajemy token, zwraca definicje sprawozdań do oddania dla usera

GET /teacher/courses
podajemy token i jeśli mamy role teacher to zwraca kursy przez niego utworzone

GET /teacher/mystudents
podajemy token i jeśli mamy role teacher to zwraca uczestników jego kursów (po uczestków jednego kursu patrz /courses/{id}/students)

GET /teacher/kickstudent/{id_kursu}/{id_studenta}
nie powinno być GETEM ale i tak wszystko podajemy w uri + token zwraca 'kicked' albo błąd

P.S. wszystkie daty podajemy w formacie

yyyy-MM-dd HH:mm
1970-01-01 00:00