

WOJSKOWA AKADEMIA TECHNICZNA

Technologie internetowe i mobilne

Temat projektu: Aplikacja to-do list

Wykonał: Jarosław Dąbrowski

Grupa: I8E2S4

Nr indeksu: 60047

1. Założenia projektu

Tematem projektu jest aplikacja "to-do list" (lista czynności do wykonania). W zakres aplikacji wchodzi następujące funkcjonalności:

- **Kategorie** : Możliwość definiowania kategorii czynności (np. praca, nauka, spotkania, rozrywka)
- **Czynności** : Możliwość dodawania czynności na osi czasu. Określenie godziny startowej i końcowej czynności. Możliwość nadania nazwy oraz opisu czynności. Możliwość odznaczenia jako wykonane.
- **Użytkownicy** : możliwość rejestracji nowych użytkowników. Każdy użytkownik może ustalać własne kategorie oraz czynności
- **Filtrowanie** : możliwość wyświetlania listy czynności o podanej kategorii
- **Sortowanie** : możliwość sortowania czynności pod względem kategorii
- **Statystyki** : możliwość wyświetlania statystyk użytkownika, np. ilość zadań w danej kategorii czy łączny czas przeznaczony na daną kategorię
- **Powiadomienia**: możliwość wysyłania powiadomień o zadaniach na adres mailowy użytkownika

2. Najważniejsze linki

Poniżej zamieszczono najważniejsze linki odnośnie projektu:

Repozytorium Github serwera i ap. mobilnej: <https://github.com/jdabrowski11926/taskManager>

Repozytorium Github ap. internetowej: <https://github.com/jdabrowski11926/taskManagerInternetApplication>

3. Serwer REST API

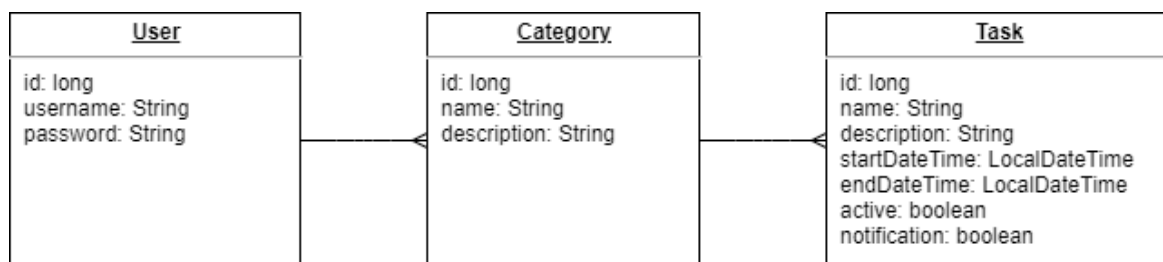
Instalacja serwera

Serwer to aplikacja napisana w języku Java przy pomocy środowiska IntelliJ IDEA w wersji 2020.1.2. W celu uruchomienia aplikacji, należy pobrać projekt z repozytorium Github a następnie zaimportować folder **“server”** do środowiska IntelliJ IDEA. Do funkcji serwera można zaliczyć:

- Przetwarzanie zapytań od zewnętrznych aplikacji zgodnie z architekturą REST,
- Autoryzacja oraz uwierzytelnianie użytkowników przy pomocy JWT (Json Web Token),
- Przechowywanie informacji o użytkownikach oraz ich kategoriach i zadaniach w postaci repozytoriów (JpaRepository),
- Wysyłanie przypomnień o zadaniach drogą mailową.

Model bazy danych

Poniżej zamieszczono model bazy danych, która zawarta jest w serwerze w postaci repozytoriów, wraz z opisami poszczególnych pól:



Rysunek 1 Model bazy danych serwera

User – użytkownik:

- id – numer id użytkownika
- Username – nazwa użytkownika. W przypadku podania w tym polu adresu email, istnieje możliwość wysyłania powiadomień drogą mailową
- Password – hasło użytkownika

Category – kategorie użytkowników:

- id – numer id kategorii
- name – nazwa kategorii. Nazwa kategorii nie może być pusta oraz nie może powtarzać się w obrębie jednego użytkownika
- description – dodatkowy opis do kategorii

Task – zadanie użytkownika:

- id – numer id zadania
- name – nazwa zadania
- description – dodatkowy opis do zadania
- startDateTime – data oraz czas rozpoczęcia zadania

- endDateTime – data oraz czas zakończenia zadania
- active – czy zadanie jest aktywne. “True” dla zadania aktywnego, “False” dla zadania nieaktywnego (ukończonego)
- notification – czy ma nastąpić powiadomienie drogą mailową. “True” – tak, “False” – nie.

Funkcje serwera

Funkcjonalność serwera można przedstawić za pomocą diagram przypadków użycia. Każdy przypadek użycia nawiązuje do ścieżki, na którą można złożyć żądanie HTTP. Poniżej przedstawiono diagram wraz z opisem każdego przypadku użycia:



Rysunek 2 Diagram przypadków użycia serwera

Nazwa, opis	1. Rejestracja – rejestracja nowego użytkownika
Ścieżka wywołania	/sign-up
Akcja	POST
Możliwe odpowiedzi	<ul style="list-style-type: none"> • 201 (Created) – użytkownik został utworzony • 406 (Not Acceptable) – w przypadku gdy użytkownik poda pustą nazwę użytkownika lub puste hasło • 409 (Conflict) – użytkownik o podanej nazwie już istnieje

Nazwa, opis	2. Logowanie – logowanie użytkownika
Ścieżka wywołania	/login
Akcja	POST
Możliwe odpowiedzi	<ul style="list-style-type: none"> • 200 (OK) – użytkownik został prawidłowo zalogowany • 401 (Unauthorized) – podano zły login lub hasło

Nazwa, opis	3. Edycja konta – zmiana hasła użytkownika
Ścieżka wywołania	/user/{username}/edit_account
Akcja	POST
Możliwe odpowiedzi	<ul style="list-style-type: none"> 200 (OK) – operacja wykonana poprawnie 401 (Unauthorized) – podano złe hasło 404 (Not Found) – użytkownik o podanej nazwie nie istnieje

Nazwa, opis	4. Pobranie listy kategorii – pobranie listy kategorii użytkownika
Ścieżka wywołania	/user/{username}/category
Akcja	GET
Możliwe odpowiedzi	<ul style="list-style-type: none"> 200 (OK) – pobranie kategorii użytkownika. Jeśli użytkownik nie istnieje zostanie zwrócona pusta lista

Nazwa, opis	5. Dodanie nowej kategorii
Ścieżka wywołania	/user/{username}/category
Akcja	POST
Możliwe odpowiedzi	<ul style="list-style-type: none"> 201 (Created) – kategoria utworzona 404 (Not Found) – użytkownik, do którego ma zostać dodana kategoria nie istnieje 406 (Not Acceptable) – nazwa kategorii nie została podana 409 (Conflict) – kategoria o podanej nazwie już istnieje

Nazwa, opis	6. Edycja kategorii - dowolna zmiana pól danej kategorii
Ścieżka wywołania	/user/{username}/category/{categoryName}
Akcja	PUT
Możliwe odpowiedzi	<ul style="list-style-type: none"> 200 (OK) – dane kategorii zostały zmienione 404 (Not Found) - kategoria nie została znaleziona

Nazwa, opis	7. Usunięcie kategorii
Ścieżka wywołania	/user/{username}/category/{categoryName}
Akcja	DELETE
Możliwe odpowiedzi	<ul style="list-style-type: none"> 200 (OK) – dane kategorii zostały zmienione 404 (Not Found) - kategoria nie została znaleziona

Nazwa, opis	8. Pobranie listy zadań kategorii – zwrócenie wszystkich zadań należących do kategorii użytkownika
Ścieżka wywołania	/user/{username}/category/{categoryName}/task
Akcja	GET
Możliwe odpowiedzi	<ul style="list-style-type: none"> 200 (OK) – pobranie listy zadań z danej kategorii. Jeżeli użytkownik lub kategoria nie istnieją – zostanie zwrócona pusta lista

Nazwa, opis	9. Dodanie nowego zadania – dodanie zadania do kategorii
Ścieżka wywołania	/user/{username}/category/{categoryName}/task
Akcja	POST
Możliwe odpowiedzi	<ul style="list-style-type: none"> • 201 (Created) – utworzenie nowego zadania • 404 (Not Found) – kategoria, do którego ma być dodane zadanie nie została znaleziona • 406 (Not acceptable) – w przypadku niepoprawnych danych (pusta nazwa, data początkowa późniejsza niż data zakończenia)

Nazwa, opis	10. Edycja zadania – dowolna zmiana pól danego zadania
Ścieżka wywołania	/user/{username}/category/{categoryName}/task/{id}
Akcja	PUT
Możliwe odpowiedzi	<ul style="list-style-type: none"> • 200 (OK) – dane zadania zostały zmienione • 404 (Not Found) – zadanie nie zostało znalezione • 406 (Not acceptable) – w przypadku niepoprawnych danych (pusta nazwa, data początkowa późniejsza niż data zakończenia)

Nazwa, opis	11. Usunięcie zadania
Ścieżka wywołania	/user/{username}/category/{categoryName}/task/{id}
Akcja	DELETE
Możliwe odpowiedzi	<ul style="list-style-type: none"> • 200 (OK) – zadanie zostało usunięte • 404 (Not Found) – zadanie nie zostało znalezione

4. Skrypt testujący

Uruchomienie skryptu

Do testowania serwera można wykorzystać skrypt **“pythonTests.py”** znajdujący się w głównym folderze z repozytorium <https://github.com/jdabrowski11926/taskManager>. Skrypt został napisany w języku Python w wersji 3.8.

Aby uruchomić skrypt trzeba mieć zainstalowany język Python. Pliki instalacyjne można znaleźć pod adresami:

- <https://www.python.org/downloads/windows/> - pliki instalacyjne dla systemów Windows
- <https://wiki.python.org/moin/BeginnersGuide/Download> - pozostałe systemy operacyjne oraz dodatkowe informacje o instalacji

W celu sprawdzenia poprawności instalacji w oknie CMD można wpisać komendę **“python –version”**, która powinna zwrócić wersję zainstalowanego Pythona.

Jeżeli instalacja przebiegła pomyślnie, można uruchomić plik testujący poprzez komendę **“PATH/pythonTests.py”**.

Zmiany kodu skryptu

Przy testowaniu może wystąpić potrzeba prostej zmiany kodu aplikacji:

- **W linijce 8** – znajduje się domyślny port 8080. Możliwe, że na komputerze na tym porcie funkcjonuje już inna aplikacja. Należy w tym miejscu ustawić ten sam port, na którym działa serwer.
- **Linijka 124 i dalej** – występują wywołania funkcji odpowiedzialne za testowanie serwera. Można w tym miejscu dodawać dodatkowe wywołania funkcji

Działanie skryptu

Skrypt odpowiada za testowanie wszystkich funkcje serwera, które zostały wymienione w rozdziale 3. Poniżej zamieszczono wywołania funkcji serwera wraz z uzyskanymi odpowiedziami.

Rejestracja

Poniżej zamieszczono komendy ze skryptu odpowiadające za utworzenie użytkowników.

- Odpowiedź 406 – podano złe dane (brak nazwy lub hasła)
- Odpowiedź 201 – użytkownik został utworzony poprawnie
- Odpowiedź 409 – użytkownik o podanej nazwie już istnieje

```
testRegisterUser("", "testPassword")
testRegisterUser("testUsername", "testPassword")
testRegisterUser("testUsername", "testPassword")
```

```
Testing registering user {http://localhost:8080/sign-up}
  Status code: 406
  Error: Not Acceptable
  Message: Username or password is empty!

Testing registering user {http://localhost:8080/sign-up}
  Status code: 201
  Body: b''

Testing registering user {http://localhost:8080/sign-up}
  Status code: 409
  Error: Conflict
  Message: Username is already taken!
```

Rysunek 3 Wynik działania skryptu - rejestracja użytkowników

Edycja użytkownika

Poniżej zamieszczono komendy ze skryptu odpowiadające za edycję użytkownika.

- Odpowiedź 404 – użytkownik o podanej nazwie nie został znaleziony
- Odpowiedź 200 – dane użytkownika zostały zmienione

```
testEditUser("testUsername2", "testPassword", "testNewPassword", token)
testEditUser("testUsername", "testPassword", "testNewPassword", token)
```

```
Testing editing user {http://localhost:8080/user/{username}/edit_account}
Status code: 404
Error: Not Found
Message: User not found!

Testing editing user {http://localhost:8080/user/{username}/edit_account}
Status code: 200
Body: b''
```

Rysunek 4 Wynik działania skryptu - edycja użytkownika

Dodawanie kategorii, wyświetlanie kategorii

Poniżej zamieszczono komendy ze skryptu odpowiadające za dodawanie oraz wyświetlanie kategorii.

- Odpowiedź 404 – użytkownik o podanej nazwie nie został znaleziony
- Odpowiedź 201 – poprawnie dodano kategorię
- Odpowiedź 200 – odpowiedź z poprawnego pobrania kategorii

```
testAddCategory("testUsername2", "Sport", "Sport activities", token)
testAddCategory("testUsername", "Sport", "Sport activities", token)
testAddCategory("testUsername", "Work", "Work at IT sector", token)
testAddCategory("testUsername", "Fun", "Meetings with friends", token)
testGetCategories("testUsername", token)
```

```
Testing adding new category {http://localhost:8080/user/{username}/category}
Status code: 404
Error: Not Found
Message: User not found!

Testing adding new category {http://localhost:8080/user/{username}/category}
Status code: 201
Body: b''

Testing adding new category {http://localhost:8080/user/{username}/category}
Status code: 201
Body: b''

Testing adding new category {http://localhost:8080/user/{username}/category}
Status code: 201
Body: b''

Testing getting all categories {http://localhost:8080/user/{username}/category}
Status code: 200
Body: b'[{"id":1,"name":"Sport","description":"Sport activities","user":"testUsername"}, {"id":2,"name":"Work","description":"Work at IT sector","user":"testUsername"}, {"id":3,"name":"Fun","description":"Meetings with friends","user":"testUsername"}]'
```

Rysunek 5 Wynik działania skryptu - dodawanie kategorii, wyświetlanie kategorii

Edycja kategorii

Poniżej zamieszczono komendy ze skryptu odpowiadające za edytowanie kategorii.

- Odpowiedź 404 – kategoria o podanej nazwie nie została znaleziona
- Odpowiedź 200 – poprawnie zmieniono dane kategorii

```
testEditCategory("testUsername", "Sport2", "Sports", "Sport (football + swimming pool)", token)
testEditCategory("testUsername", "Sport", "Sports", "Sport (football + swimming pool)", token)
```

```
Testing editing category {http://localhost:8080/user/{username}/category/{categoryName}}
Status code: 404
Error: Not Found
Message: Category not found!

Testing editing category {http://localhost:8080/user/{username}/category/{categoryName}}
Status code: 200
Body: b''
```

Rysunek 6 Wynik działania skryptu - edycja kategorii

Usuwanie kategorii

Poniżej zamieszczono komendy ze skryptu odpowiadające za usuwanie kategorii.

- Odpowiedź 404 – kategoria o podanej nazwie nie została znaleziona
- Odpowiedź 200 – poprawnie usunięto kategorię
- Odpowiedź 200 – ponowne pobranie kategorii. Można zaobserwować, że w porównaniu z poprzednim pobraniem kategorii, jedna kategoria została usunięta, a druga została zmieniona

```
testDeleteCategory("testUsername", "Fun2", token)
testDeleteCategory("testUsername", "Fun", token)
testGetCategories("testUsername", token)
```

```
Testing deleting category {http://localhost:8080/user/{username}/category/{categoryName}}
Status code: 404
Error: Not Found
Message: category not found!

Testing deleting category {http://localhost:8080/user/{username}/category/{categoryName}}
Status code: 200
Body: b''

Testing getting all categories {http://localhost:8080/user/{username}/category}
Status code: 200
Body: b'[{"id":1,"name":"Sports","description":"Sport (football + swimming pool)","user":"testUsername"}, {"id":2,"name":"Work","description":"Work at IT sector","user":"testUsername"}]'
```

Rysunek 7 Wynik działania skryptu - usuwanie kategorii

Dodawanie zadań do kategorii oraz wyświetlanie zadań

Poniżej zamieszczono komendy ze skryptu odpowiadające za dodawanie zadań do kategorii.

- Odpowiedź 404 - kategoria o podanej nazwie nie została znaleziona
- Odpowiedź 406 – niepoprawne dane. Data początkowa zadania jest późniejsza niż data zakończenia zadania
- Odpowiedź 201 – zadanie zostało dodane poprawnie
- Odpowiedź 200 – wyświetlenie dodanych zadań

```
testAddTask("testUsername2","Sports","Swimming pool","Swimming pool with friends",
    "01/06/2020 12:00","01/06/2020 14:00",True,False, token)
testAddTask("testUsername","Sports","Swimming pool","Swimming pool with friends",
    "01/06/2020 16:00","01/06/2020 14:00",True,False, token)
testAddTask("testUsername","Sports","Swimming pool","Swimming pool with friends",
    "01/06/2020 12:00","01/06/2020 14:00",True,False, token)
testAddTask("testUsername","Sports","Football","Football with friends",
    "02/06/2020 14:00","02/06/2020 16:00",True,False, token)
testAddTask("testUsername","Sports","Badminton","Badminton with coach",
    "03/06/2020 14:00","03/06/2020 16:00",True,False, token)
testAddTask("testUsername","Sports","Badminton","Badminton with coach",
    "03/06/2020 15:00","03/06/2020 17:00",True,False, token)
getTasks("testUsername","Sports", token)
```

```
Testing adding task {http://localhost:8080/user/{username}/category/{categoryName}/task}
Status code: 404
Error: Not Found
Message: Category not found!

Testing adding task {http://localhost:8080/user/{username}/category/{categoryName}/task}
Status code: 406
Error: Not Acceptable
Message: Inserted invalid data, start date is later than end date

Testing adding task {http://localhost:8080/user/{username}/category/{categoryName}/task}
Status code: 201
Body: b''

Testing adding task {http://localhost:8080/user/{username}/category/{categoryName}/task}
Status code: 201
Body: b''

Testing adding task {http://localhost:8080/user/{username}/category/{categoryName}/task}
Status code: 201
Body: b''

Testing adding task {http://localhost:8080/user/{username}/category/{categoryName}/task}
Status code: 201
Body: b''

Testing getting task {http://localhost:8080/user/{username}/category/{categoryName}/task}
Status code: 200
Body: b'{"id":1,"name":"Swimming pool","description":"Swimming pool with friend
s","startDateTime":"01/06/2020 12:00","endDateTime":"01/06/2020 14:00","active":true,"not
ification":false,"categoryId":1},{\"id\":2,\"name\":\"Football\",\"description\":\"Football with f
riends\",\"startDateTime\":\"02/06/2020 14:00\",\"endDateTime\":\"02/06/2020 16:00\",\"active\":true
,\"notification\":false,\"categoryId\":1},{\"id\":3,\"name\":\"Badminton\",\"description\":\"Badminkt
on with coach\",\"startDateTime\":\"03/06/2020 14:00\",\"endDateTime\":\"03/06/2020 16:00\",\"activ
e\":true,\"notification\":false,\"categoryId\":1},{\"id\":4,\"name\":\"Badminton\",\"description\":\"B
adminton with coach\",\"startDateTime\":\"03/06/2020 15:00\",\"endDateTime\":\"03/06/2020 17:00\"
,\"active\":true,\"notification\":false,\"categoryId\":1}]'
```

Rysunek 8 Wynik działania skryptu - dodawanie zadań do kategorii oraz wyświetlanie zadań

Edytowanie zadań

Poniżej zamieszczono komendy ze skryptu odpowiadające za edytowanie zadań.

- Odpowiedź 404 - zadanie nie zostało znalezione
- Odpowiedź 200 – zadanie zostało poprawnie edytowane

```
testEditTask("testUsername","Sports2", 2, "Football", "Football with team",
    "01/06/2020 15:00","01/06/2020 18:00",True,False, token)
testEditTask("testUsername","Sports", 2, "Football", "Football with team",
    "01/06/2020 15:00","01/06/2020 18:00",True,False, token)
```

```
Testing editing task {http://localhost:8080/user/{username}/category/{categoryName}/task/{taskId}}
Status code: 404
Error: Not Found
Message: Task not found

Testing editing task {http://localhost:8080/user/{username}/category/{categoryName}/task/{taskId}}
Status code: 200
Body: b''
```

Rysunek 9 Wynik działania skryptu - edytowanie zadań

Usuwanie zadań

Poniżej zamieszczono komendy ze skryptu odpowiadające za usuwanie zadań.

- Odpowiedź 404 - zadanie nie zostało znalezione
- Odpowiedź 200 – zadanie zostało poprawnie usunięte
- Odpowiedź 200 – poprawne wyświetlenie zadań po usunięciu oraz edytowaniu

```
testDeleteTask("testUsername","Sports2", 3, token)
testDeleteTask("testUsername","Sports", 3, token)
getTasks("testUsername","Sports", token)
```

```
Testing deleting task {http://localhost:8080/user/{username}/category/{categoryName}/task/{taskId}}
Status code: 404
Error: Not Found
Message: Task not found

Testing deleting task {http://localhost:8080/user/{username}/category/{categoryName}/task/{taskId}}
Status code: 200
Body: b''

Testing getting task {http://localhost:8080/user/{username}/category/{categoryName}/task}
Status code: 200
Body: b'[{"id":1,"name":"Swimming pool","description":"Swimming pool with friends","startDateTime":"01/06/2020 12:00","endDateTime":"01/06/2020 14:00","active":true,"notification":false,"categoryId":1},{"id":2,"name":"Football","description":"Football with team","startDateTime":"01/06/2020 15:00","endDateTime":"01/06/2020 18:00","active":true,"notification":false,"categoryId":1},{"id":4,"name":"Badminton","description":"Badminton with coach","startDateTime":"03/06/2020 15:00","endDateTime":"03/06/2020 17:00","active":true,"notification":false,"categoryId":1}]'
```

Rysunek 10 Wynik działania skryptu - usuwanie zadań

5. Aplikacja mobilna

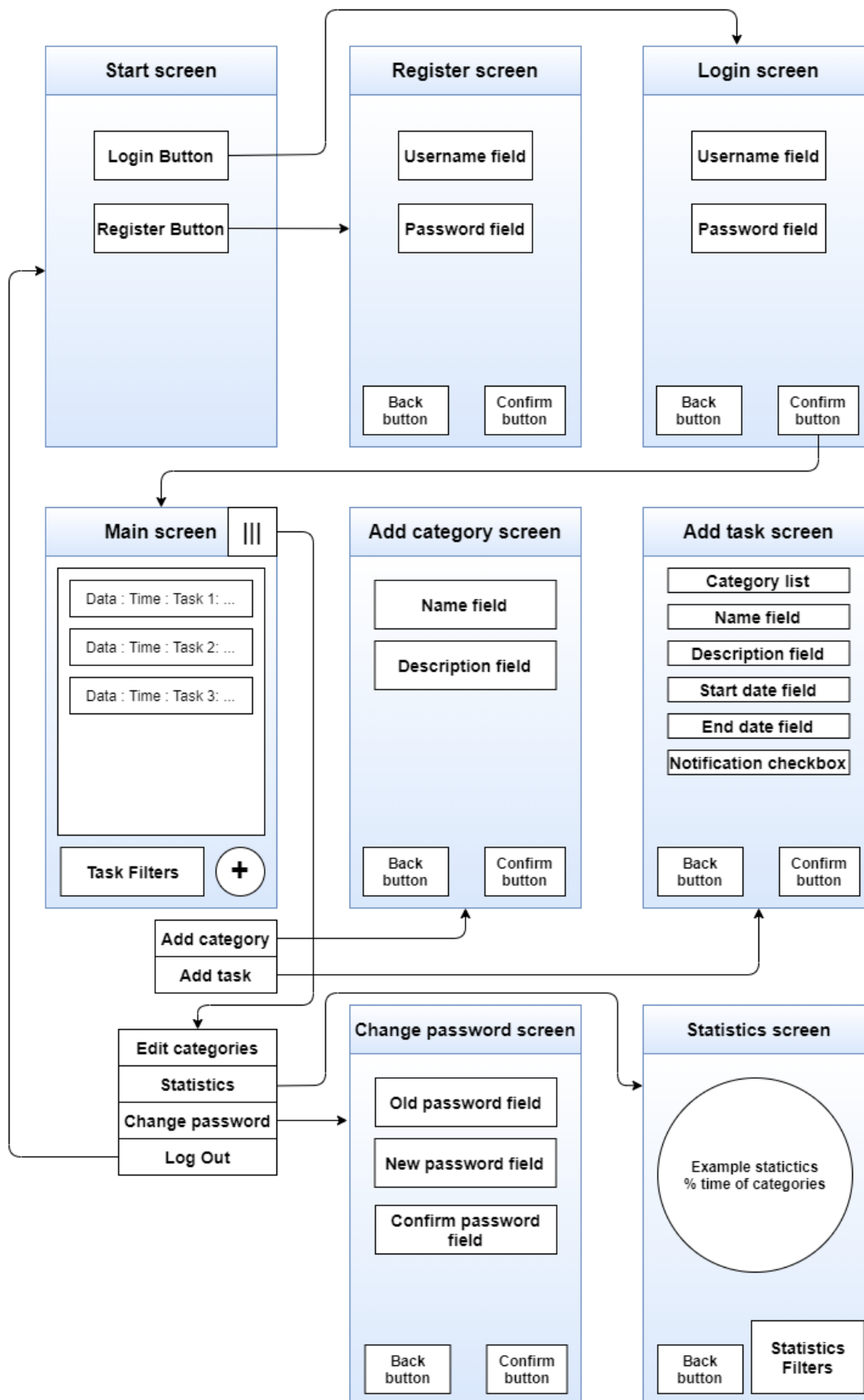
Uruchomienie aplikacji

Aplikacja mobilna została napisana w języku Kotlin przy pomocy środowiska Android Studio w wersji 3.6.3. W celu uruchomienia aplikacji, należy pobrać projekt z repozytorium Github a następnie zaimportować folder **“mobileClient”** do środowiska Android Studio.

Aby aplikacja działała poprawnie, uprzednio musi być uruchomiona aplikacja **“server”** w środowisku Intelij IDEA. W przeciwnym wypadku nie będzie możliwości zarejestrowania użytkowników oraz przejścia do kolejnych ekranów.

Funkcje oraz model aplikacji

Aplikacja mobilna stanowi interfejs komunikacji z użytkownikiem, dzięki któremu można wykorzystać wszystkie funkcje serwera. Poniżej przedstawiono schemat ekranów aplikacji, wraz z możliwymi przejściami pomiędzy nimi

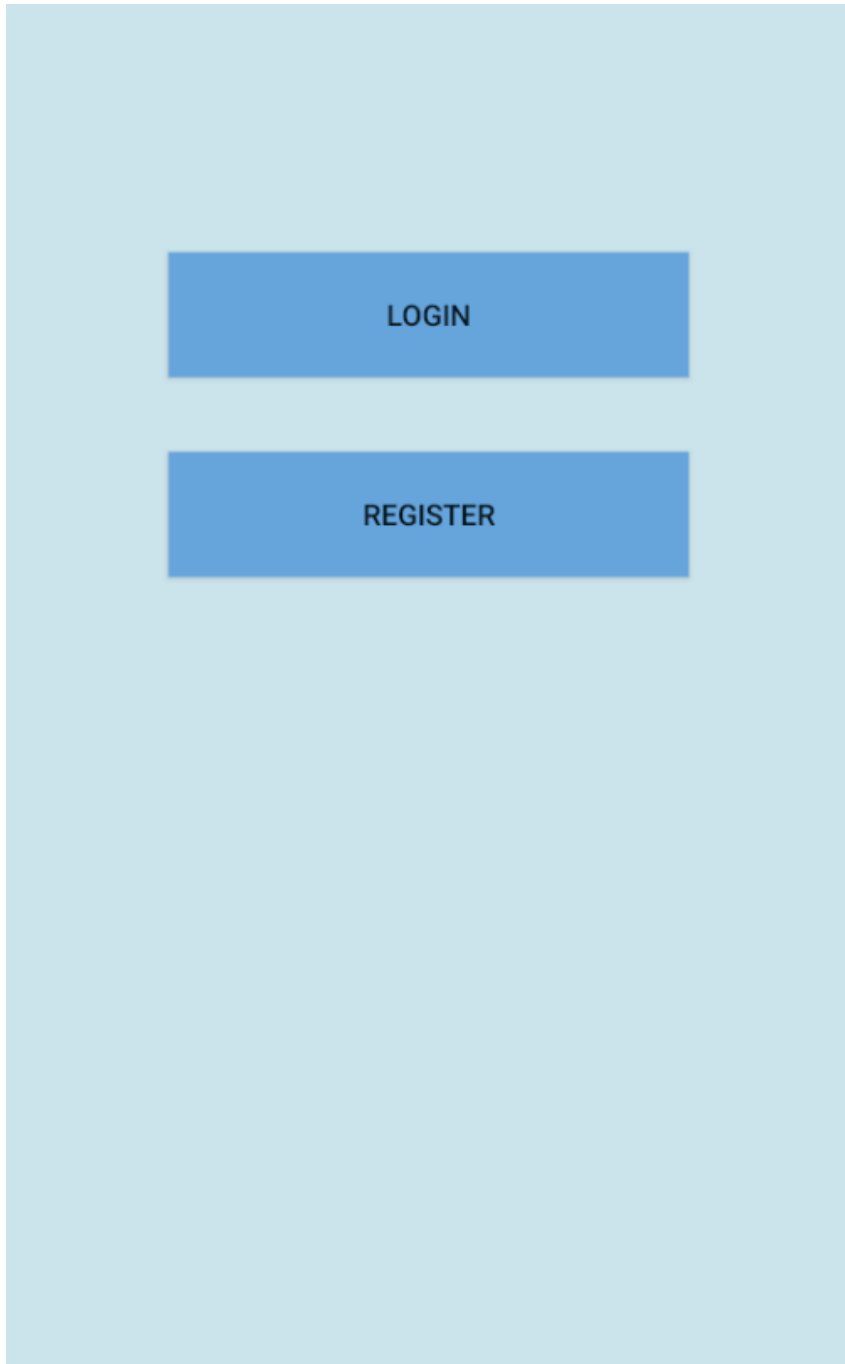


Rysunek 11 Model ekranów aplikacji mobilnej

Wynik implementacji

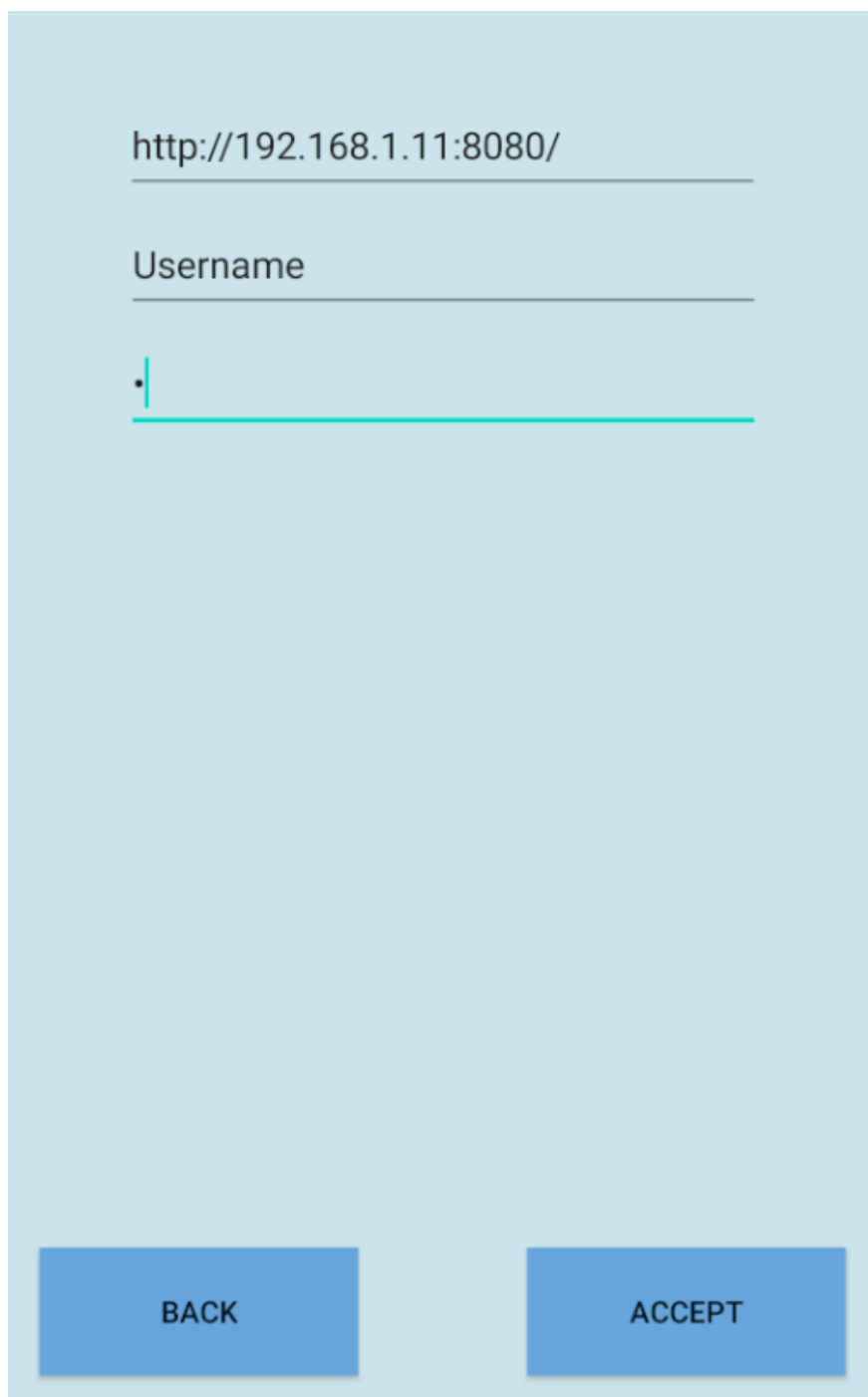
W rozdziale tym przedstawiono przykładowe wykorzystanie aplikacji, zrzuty ekranów oraz opis funkcjonalności.

Ekran startowy – ekran ukazujący się po uruchomieniu aplikacji. Z tego miejsca można przejść do rejestracji oraz logowania użytkownika.



Rysunek 12 Aplikacja mobilna - ekran startowy

Ekran logowania i rejestracji – pod kątem graficznym oba ekrany wyglądają identycznie. Różnią się funkcją przypisaną do przycisku „ACCEPT”. Przy rejestracji następuje żądanie do serwera z utworzeniem nowego użytkownika, przy logowaniu następuje sprawdzenie, czy dany użytkownik rzeczywiście znajduje się w bazie danych. Nad nazwą oraz hasłem użytkownika znajduje się pole z nazwą serwera. Można ją zmienić w przypadku gdy serwer funkcjonuje na innym porcie niż prezentowany w przykładzie.



http://192.168.1.11:8080/

Username

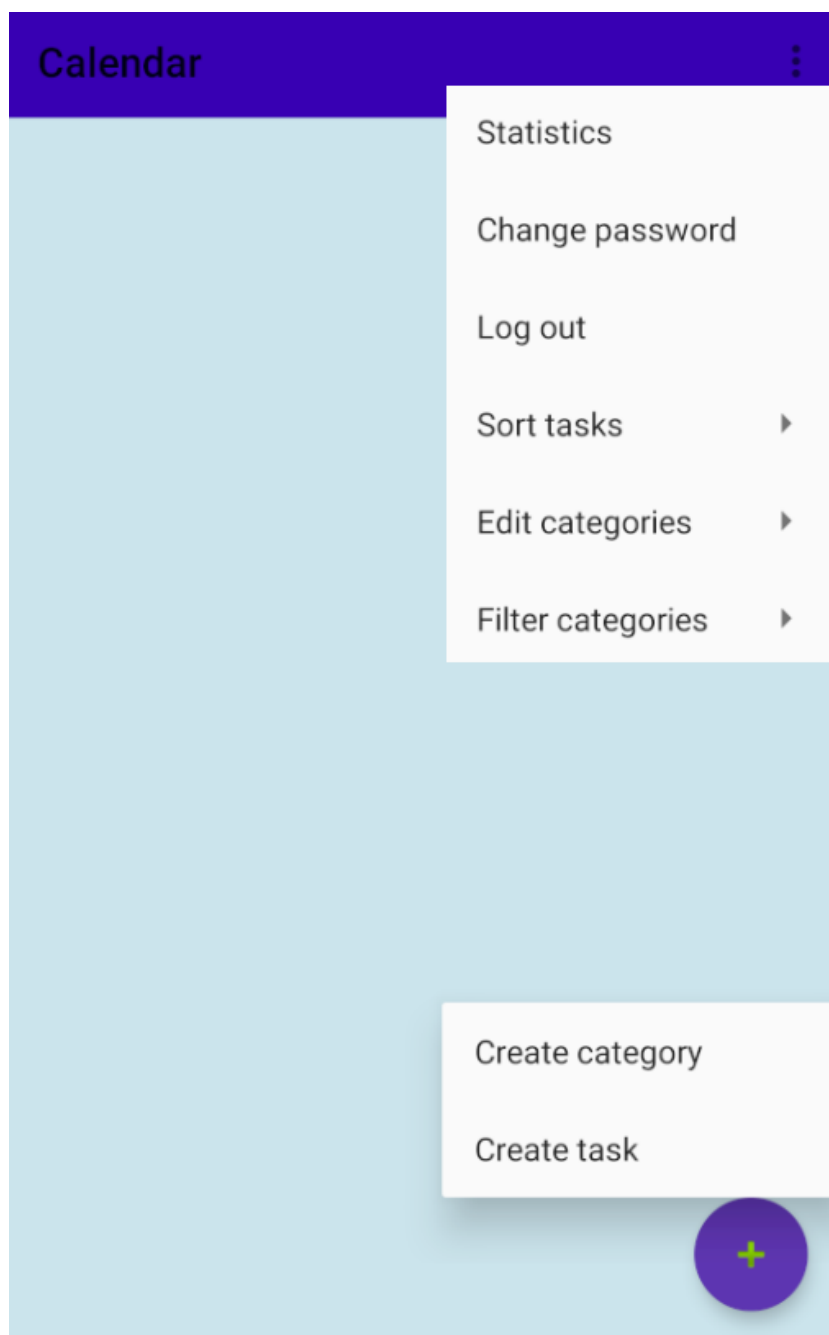
•

BACK ACCEPT

Rysunek 13 Aplikacja mobilna - ekran rejestracji i logowania

Ekran kalendarza – główny ekran zawierający wszystkie zadania przypisane do użytkownika. W górnym prawym rogu znajduje się menu opcji, umożliwiające wylogowanie (**Log Out**), przejście do innych ekranów (**Statistics, Change password, Edit categories**) oraz sortowanie oraz filtrowanie obecnie wyświetlanych zadań (**sort tasks, filter categories**).

Początkowo użytkownik nie posiada jednak żadnych zadań, dlatego zaleca się najpierw dodać nową kategorię a następnie dodać nowe zadania. Można to zrobić za pośrednictwem przycisku w dolnym prawym rogu (**Create category, Create task**).



Rysunek 14 Aplikacja mobilna - ekran kalendarza

Ekran kategorii – w aplikacji występują dwa ekrany kategorii. Ekran dodawania kategorii oraz ekran edycji kategorii. Pod kątem graficznym są one identyczne, ale różnią się funkcją przycisku „**ACCEPT**”. Przy nowej kategorii wysyłane jest żądanie do serwera o przypisanie nowej kategorii, dodatkowo przycisk usunięcia kategorii „**DELETE**” jest zablokowany (nie można usunąć kategorii, która jeszcze nie istnieje). Przy edycji kategorii przycisk „**DELETE**” jest dostępny, a przycisk „**ACCEPT**” odpowiada za wysłanie żądania edycji kategorii do serwera.



Category name Sport

Description Wszystkie zadania o sporcie

BACK DELETE ACCEPT

Rysunek 15 Aplikacja mobilna - ekran kategorii

Ekran zadania – podobnie jak przy kategoriach, w aplikacji występują dwa ekrany zadań. Ekran dodawania zadania oraz ekran edycji zadania. Przy nowym zadaniu nie ma możliwości jego usunięcia, akceptacja wysyła żądanie do serwera o przypisanie nowej kategorii. Przy edycji zadania można usunąć zadanie, a akceptacja wysyła żądanie edycji.

Należy w tym miejscu zwrócić uwagę na pole kategorii. Przy ładowaniu ekranu następuje żądanie z serwera o pobranie listy kategorii, która jest ładowana do listy z możliwością wyboru. Jeżeli użytkownik nie podał wcześniej żadnej kategorii – nie będzie mógł on wysłać zadania za pośrednictwem przycisku „ACCEPT”.

Dodatkowo, po dwukrotnym wciśnięciu pól przeznaczonych na daty oraz czasy – wyświetlą się okna ułatwiające wybór wartości.

Category	Sport	▼
Name	Basen	
Description	Basen ze znajomymi	
Start date/time	10/06/2020	13:30
End date/time	10/06/2020	14:30
Notification on email?	<input checked="" type="checkbox"/>	
Is active?	<input checked="" type="checkbox"/>	

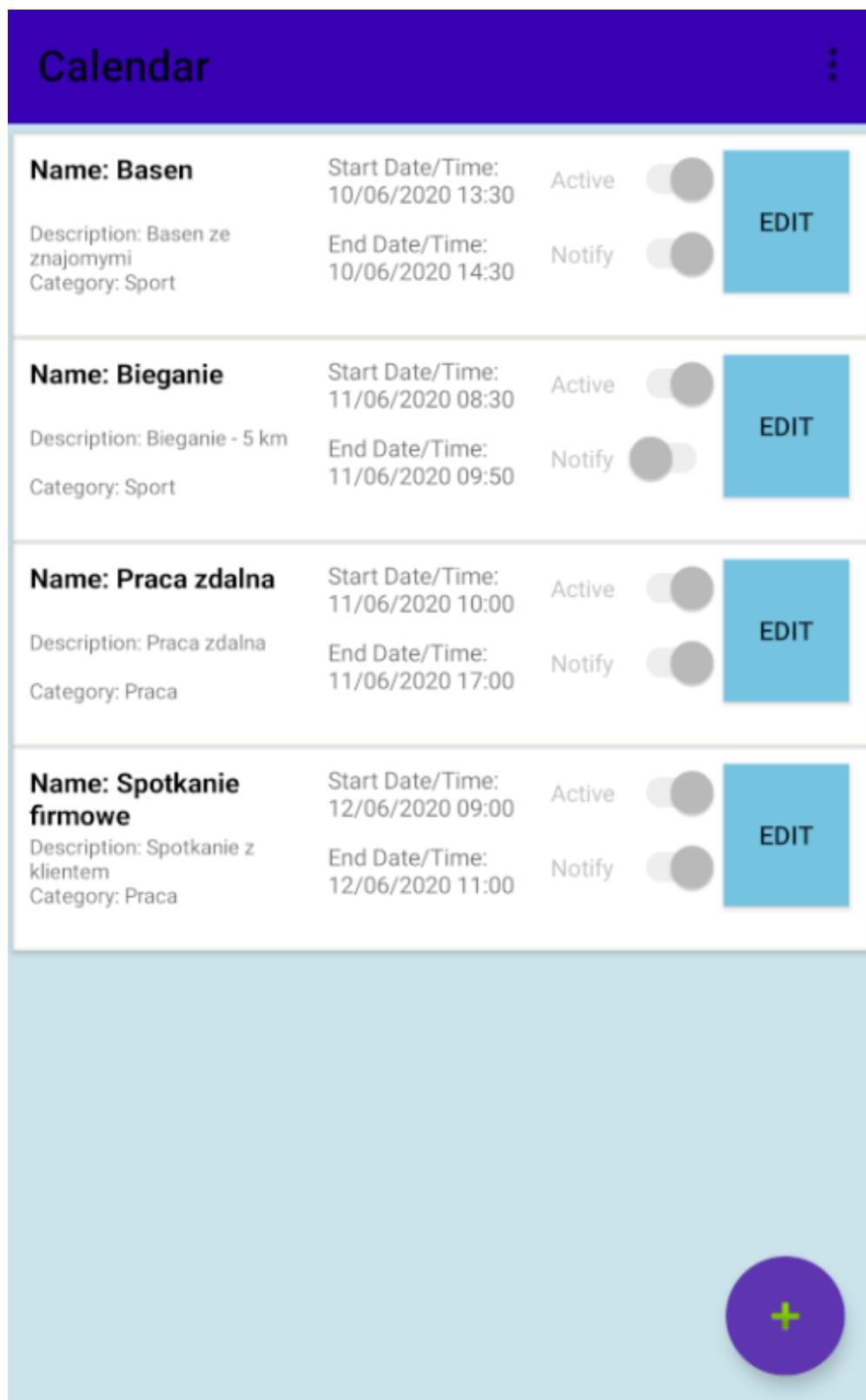
BACK

DELETE

ACCEPT

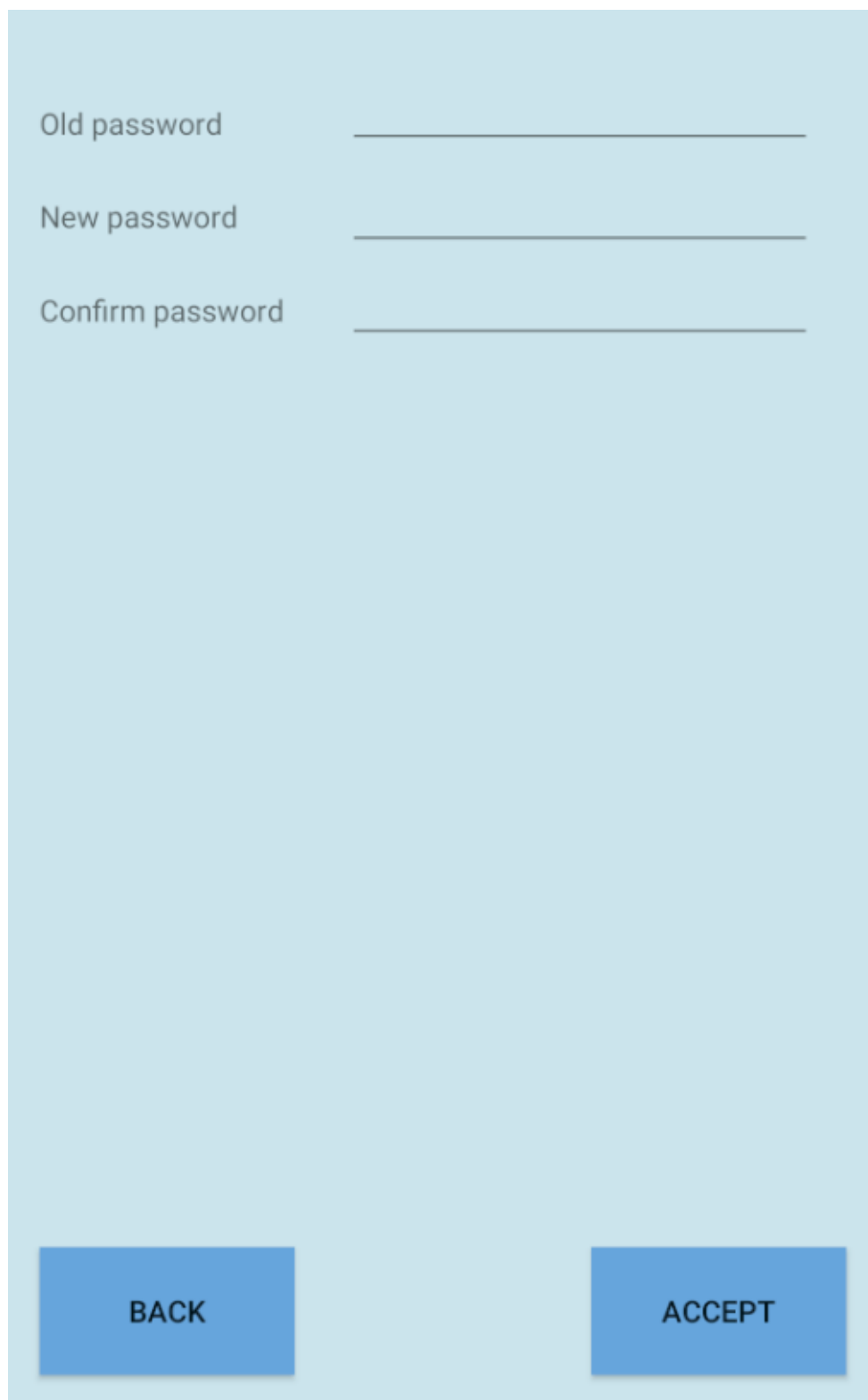
Rysunek 16 Aplikacja mobilna - ekran zadania

Ekran kalendarza (2) – przy ładowaniu ekranu głównego kalendarza następuje żądanie do serwera z pobraniem wszystkich zadań należących do użytkownika. Będą one zatem na bieżąco ładowane przy każdym dodaniu nowego zadania. W tym miejscu można już wypróbować opcje filtrowania oraz sortowania zadań (które znajdują się w opcjach w górnej prawej stronie ekranu).



Rysunek 17 Aplikacja mobilna - ekran kalendarza po dodaniu zadań

Ekran zmiany hasła – ekran odpowiedzialny za edycję użytkownika w postaci zmiany jego hasła. Aby je zmienić należy podać stare hasło oraz dwukrotnie podać nowe hasło (New password, confirm password). Po wciśnięciu przycisku „ACCEPT” następuje sprawdzenie czy nowe hasło zostało podane dwukrotnie tak samo, jeśli tak to następuje wysłanie prośby do serwera o edycję hasła.



Old password

New password

Confirm password

BACK ACCEPT

Rysunek 18 Aplikacja mobilna - ekran zmiany hasła

Ekran statystyk – ekran przedstawiający statystyki użytkownika. Statystyki przedstawione są w postaci tabeli o kolumnach:

- Category name – nazwa kategorii
- Amount of tasks – liczba zadań przypisanych do kategorii
- Time for All tasks – łączny czas poświęcony na wszystkie zadania z danej kategorii

Category name	Amount of tasks	Time for all tasks
Sport	2	2 hrs, 20 min, 0sec
Praca	2	9 hrs, 0 min, 0sec

BACK

Rysunek 19 Aplikacja mobilna - ekran statystyk

6. Aplikacja internetowa

Uruchomienie aplikacji

Aplikacja mobilna została napisana środowisku React.js przy wykorzystaniu języka Javascript oraz HTML oraz CSS do wyświetlania grafiki.

Do aplikacji w React.js wymagane jest środowisko Node.js. Pliki instalacyjne można pobrać ze strony <https://nodejs.org/en/download/>. W celu weryfikacji instalacji, można wpisać w CMD komendę **"npm -v"**, która zwróci wersję Node.js (wersja, na której została napisana aplikacja to 6.14.5).

Następnie aby zainstalować React.js, można skorzystać z następujących komend:

- **npm install -g create-react-app** – komenda do instalacji React.js
- **create-react-app -- version** – jeżeli instalacja przebiegła prawidłowo, komenda ta powinna zwrócić wersję zainstalowanego React.js (wersja, na której została napisana aplikacja to 3.4.1)
- **cd <projectname>** - zmiana folderu na folder zawierający projekt
- **npm start** – uruchomienie aplikacji

Więcej informacji o komendach instalacyjnych można znaleźć np. Pod linkiem <https://makandracards.com/reactjs-quick/52419-install-reactjs-windows>.

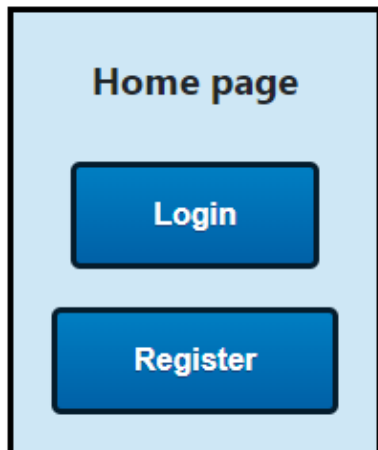
W celu uruchomienia aplikacji projektowej, należy pobrać projekt z repozytorium Github (<https://github.com/jdabrowski11926/taskManagerInternetApplication>), zmienić obecną ścieżkę przy pomocy komendy **cd <projectname>** a następnie uruchomić przy pomocy komendy **npm start**.

Aby aplikacja działała poprawnie, uprzednio musi być uruchomiona aplikacja **"server"** w środowisku Intelij IDEA. W przeciwnym wypadku nie będzie możliwości zarejestrowania użytkowników oraz przejścia do kolejnych ekranów.

Wynik implementacji

W rozdziale tym przedstawiono przykładowe wykorzystanie aplikacji internetowej, zrzuty ekranów oraz opis funkcjonalności.

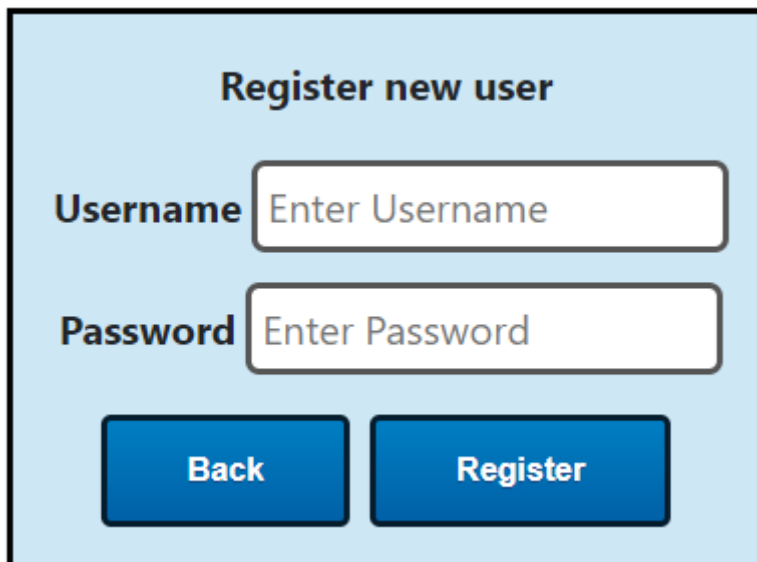
Ekran startowy - ekran ukazujący się po uruchomieniu aplikacji. Z tego miejsca można przejść do rejestracji oraz logowania użytkownika.



The screenshot shows a light blue rectangular area with a black border. At the top, the text "Home page" is centered in a bold, black font. Below this, there are two blue rectangular buttons with black borders, stacked vertically. The top button is labeled "Login" in white text, and the bottom button is labeled "Register" in white text.

Rysunek 20 Aplikacja internetowa - ekran startowy

Ekran rejestracji i logowania - pod kątem graficznym oba ekrany wyglądają identycznie. Różnią się funkcją przypisaną do przycisku. Przy rejestracji następuje żądanie do serwera o zapisanie nowego użytkownika. Przy logowaniu następuje żądanie do serwera o sprawdzenie, czy dany użytkownik rzeczywiście istnieje. Jeżeli dane się zgadzają następuje przejście do ekranu kalendarza.



The screenshot shows a light blue rectangular area with a black border. At the top, the text "Register new user" is centered in a bold, black font. Below this, there are two input fields. The first is labeled "Username" in bold black text, followed by a white rectangular box with a black border containing the placeholder text "Enter Username". The second is labeled "Password" in bold black text, followed by a white rectangular box with a black border containing the placeholder text "Enter Password". At the bottom, there are two blue rectangular buttons with black borders. The left button is labeled "Back" in white text, and the right button is labeled "Register" in white text.

Rysunek 21 Aplikacja internetowa - ekran rejestracji i logowania

Ekran kalendarza – po poprawnym zalogowaniu następuje przejście do ekranu kalendarza. Przy jego ładowaniu do serwera wysyłane jest zapytanie o listę zadań przypisanych do użytkownika.

Jeżeli wcześniej do użytkownika przypisano zadania za pośrednictwem aplikacji mobilnej, a serwer nie został jeszcze wyłączony, będzie można zauważyć te same zadania w aplikacji internetowej (tak jak zostało to zaprezentowane na obrazku poniżej).

Calendar site

New Category

New Task

Change password

Task Name	Basen	Start date/time	10/06/2020 13:30
Task description	Basen ze znajomymi	End date/time	10/06/2020 14:30
Task category	Task category	Active	<input checked="" type="checkbox"/> Notification <input checked="" type="checkbox"/>
Task Name	Bieganie	Start date/time	11/06/2020 08:30
Task description	Bieganie - 5 km	End date/time	11/06/2020 09:50
Task category	Task category	Active	<input checked="" type="checkbox"/> Notification <input type="checkbox"/>
Task Name	Praca zdalna	Start date/time	11/06/2020 10:00
Task description	Praca zdalna	End date/time	11/06/2020 17:00
Task category	Task category	Active	<input checked="" type="checkbox"/> Notification <input checked="" type="checkbox"/>
Task Name	Spotkanie firmowe	Start date/time	12/06/2020 09:00
Task description	Spotkanie z klientem	End date/time	12/06/2020 11:00
Task category	Task category	Active	<input checked="" type="checkbox"/> Notification <input checked="" type="checkbox"/>

Rysunek 22 Aplikacja internetowa - ekran kalendarza

Ekran zmiany hasła - ekran odpowiedzialny za edycję użytkownika w postaci zmiany jego hasła. Aby je zmienić należy podać stare hasło oraz dwukrotnie podać nowe hasło. Po wciśnięciu przycisku „Apply” następuje sprawdzenie czy nowe hasło zostało podane dwukrotnie tak samo, jeśli tak to następuje wysłanie prośby do serwera o edycję hasła.

The screenshot shows a web form titled "Change Password" on a light blue background. It contains three input fields: "Old Password" with placeholder text "Enter old Password", "New Password" with placeholder text "Enter new Password", and "Confirm new password" with placeholder text "Confirm Password". Below the input fields are two blue buttons: "Back" and "Apply".

Rysunek 23 Aplikacja internetowa - ekran zmiany hasła

Ekran nowej kategorii – ekran umożliwiający wysłanie zapytań do serwera o przypisywanie nowych kategorii do użytkownika. Akcja następuje po wciśnięciu przycisku „Apply”.

The screenshot shows a web form titled "New category" on a light blue background. It contains two input fields: "Category name" with placeholder text "Enter category name" and "Category description" with placeholder text "Enter description name". Below the input fields are two blue buttons: "Back" and "Apply".

Rysunek 24 Aplikacja internetowa - ekran nowej kategorii

Ekran nowego zadania – ekran umożliwiający dodawanie nowych zadań do kategorii użytkownika. Przy ładowaniu tego ekranu następuje zapytanie do serwera o listę kategorii, która jest ładowana do listy z wyborem. Kategoria jest polem obowiązkowym, więc jeżeli użytkownik nie dodał wcześniej żadnych kategorii, nie będzie możliwości przypisania zadania.

New task

Category Sport

Task name Enter task name

Task description Enter task description

Task start date/time dd.mm.rrrr --:--

Task end date/time dd.mm.rrrr --:--

Active ☐

Notification ☐

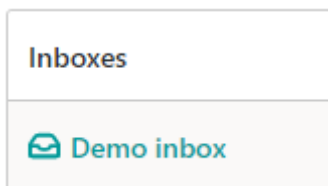
BackApply

7. Powiadomienia drogą mailową

Za wysyłanie powiadomień drogą mailową odpowiedzialny jest serwer, jednak ich otrzymywanie wymaga krótkiej konfiguracji.

Do testowania powiadomień drogą mailową wykorzystany został serwis <https://mailtrap.io>, który pozwala na wysyłanie powiadomień testowych. Można się na niego zalogować przy pomocy popularnych serwisów jak Google czy Github.

Po zalogowaniu należy przejść w skrzynkę "Demo inbox".



Rysunek 25 Mailtrap - demo inbox

W tym miejscu można znaleźć dane dotyczące protokołu komunikacyjnego SMTP, które powinny być wyświetlone po lewej stronie ekranu.

Credentials [Reset SMTP/POP3](#)

SMTP

Host: smtp.mailtrap.io
Port: 25 or 465 or 587 or 2525
Username: 8c8a4871d74236
Password: c3414a0b57f18d
Auth: PLAIN, LOGIN and CRAM-MD5
TLS: Optional (STARTTLS on all ports)

Rysunek 26 Mailtrap - dane SMTP

Aby serwer wysyłał dane do testowych skrzynek odbiorczych, należy skopiować pola: Host, Port (dowolny), Username oraz Password, a następnie wkleić je do ustawień serwera. Ustawienia serwera można znaleźć pod ścieżką src/main/java/WAT/I8E2S4/TaskManager/security. Poniżej przedstawiono kod serwera, który powinien być zmieniony.

```
public static final String MAIL_HOST = "smtp.mailtrap.io";  
public static final int MAIL_PORT = 587;  
public static final String MAIL_USERNAME = "8c8a4871d74236";  
public static final String MAIL_PASSWORD = "c3414a0b57f18d";
```

Rysunek 27 Kod serwera - ustawienia skrzynki mailowej

Po dokonaniu powyższych zmian na skrzynkę <https://mailtrap.io> powinny przychodzić wiadomości przypominające o zadaniach. Poniżej zamieszczono przykładowe powiadomienie:

Przypomnienie o zadaniu

From: <taskControllerTIM@gmail.com>

To: <TestUser@gmail.com>

[Show Info](#)

HTML	HTML Source	Text	Raw	Analysis	SMTP info
------	-------------	------	-----	----------	-----------

Na 2020-06-11T10:47 - 2020-06-11T11:46 zostało zaplanowane zadanie: Sport(opis : Sport ze znajomymi)

Rysunek 28 Przykładowe powiadomienie drogą mailową