Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра програмної забезпечення

Магістерська кваліфікаційна робота

на тему: Розробка методів реактивного виведення даних для веб-сервісу зберігання та відтворення аудіофайлів

ВИКОНАВ:
студент групи 1ПІ-18м Корягін І. С.

НАУКОВИЙ КЕРІВНИК:
к.т.н., доцент кафедри ПЗ Ракитянська Г.Б.

Мета, предмет та об’єкт дослідження

Мета та завдання дослідження – підвищення продуктивності веб-сервісу зберігання та відтворення аудіофайлів за рахунок реактивного виведення даних.

Об'єктом дослідження є процес реактивного виведення даних.

Предметом дослідження є методи і засоби реактивного виведення даних.

Актуальність теми
Низька продуктивність веб-сервісу

Час очікування

При навігації по веб-сервісу час очікування появи контенту сторінки занадто великий

Надмірна кількість запитів

Сервіс завантажує декілька разів однакові дані на різних вкладках(наприклад, список жанрів, категорій)

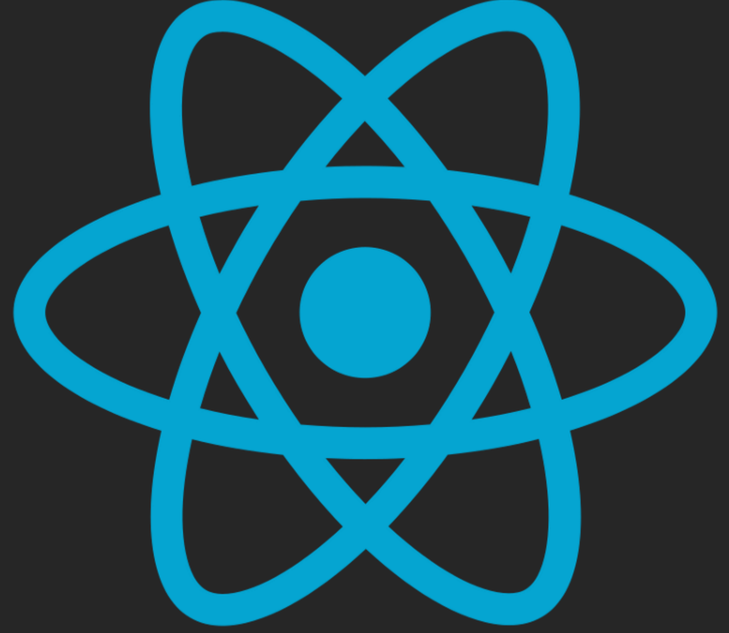
Наукова новизна отриманих результатів

1. Вперше запропоновано метод реактивного виведення даних, особливість якого полягає у збереженні даних в проміжному сховищі у вигляді структурованого об’єкту, що забезпечує зменшення кількості запитів до серверу.

2. Подальшого розвитку отримав метод селективного вибору отриманих даних, у якому, на відміну існуючих, відслідковуються зміни даних проміжного сховища, що забезпечує автоматичне оновлення вибраних даних.

Переваги Angular:

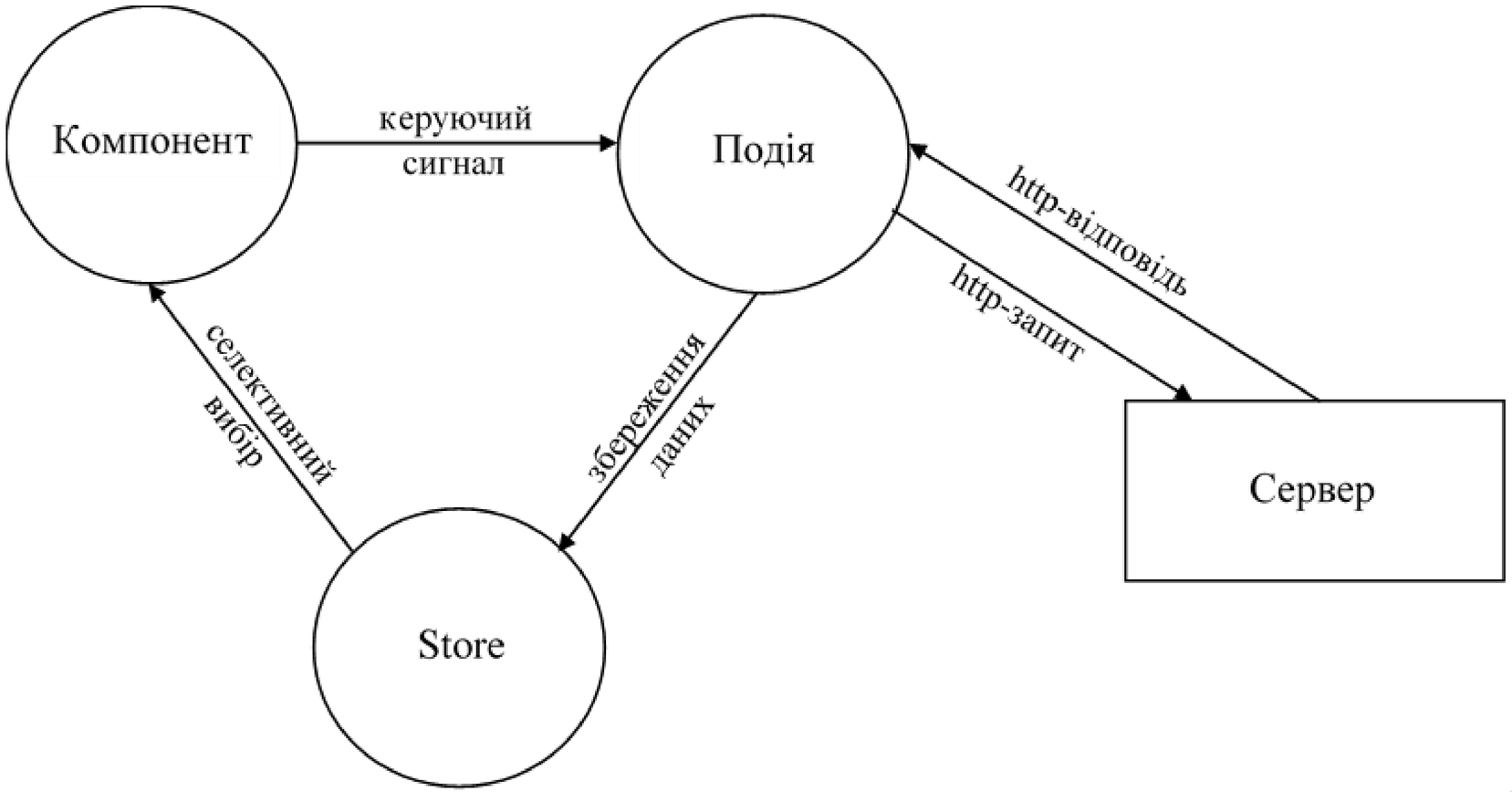
- Підтримка веб-компонентів.
- Швидкість роботи.
- Відмінна продуктивність.
- Використання Typescript.



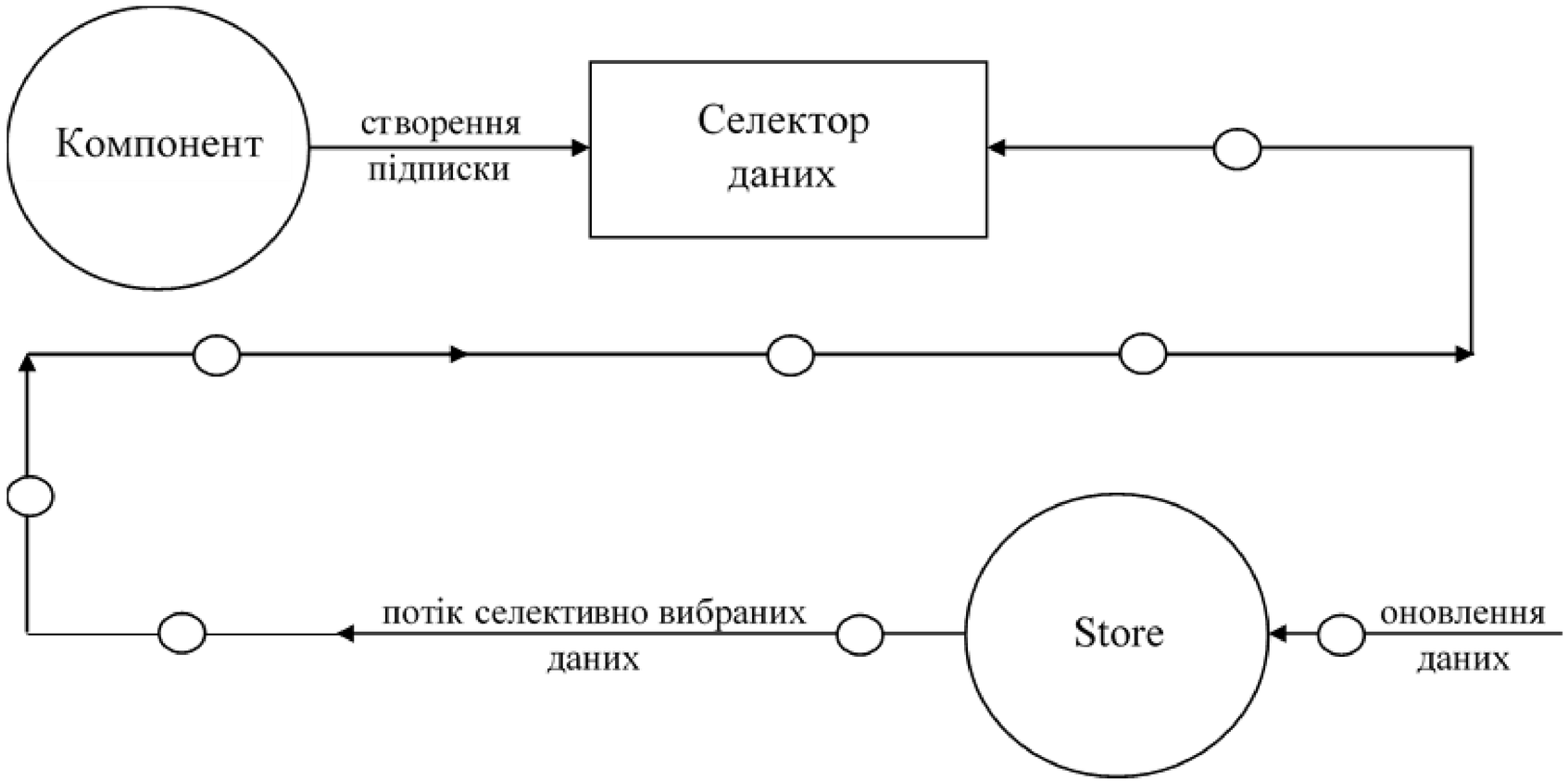
Методи реалізації

Angular представляє фреймворк від компанії Google для створення клієнтських додатків. Перш за все він націлений на розробку SPA-рішень (Single Page Application), тобто односторінкових додатків.

Архітектура методу збереження даних до проміжного сховища Store



Архітектура методу потокового виведення даних

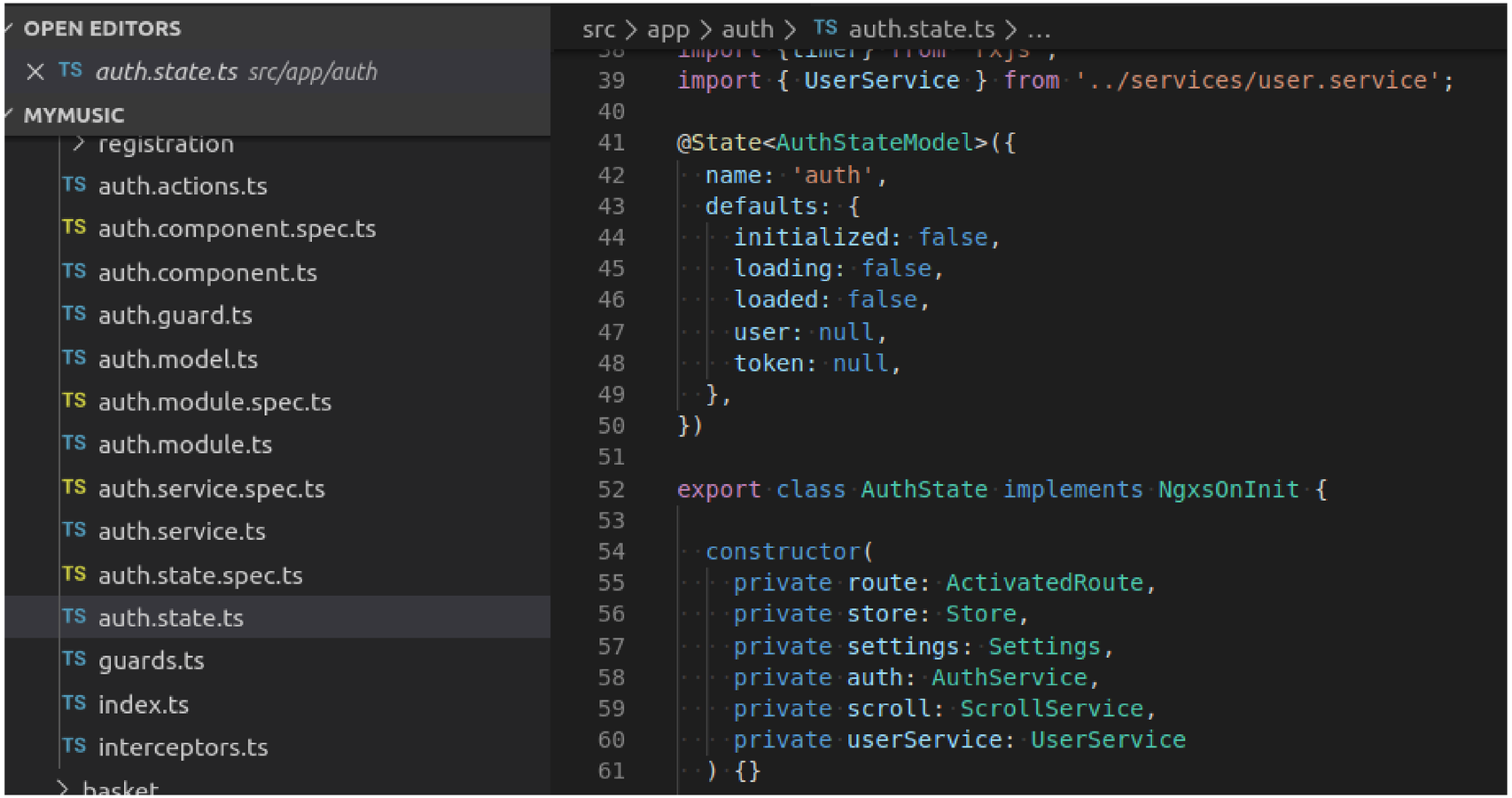


В компоненті веб-сервісу створюється підписка на селектор даних зі сховища даних Store, яка відслідковує зміни в екземплярі сховища і при оновленні даних, отриманих після відповіді на http-запит, автоматично додає їх в потік на який підписався компонент.

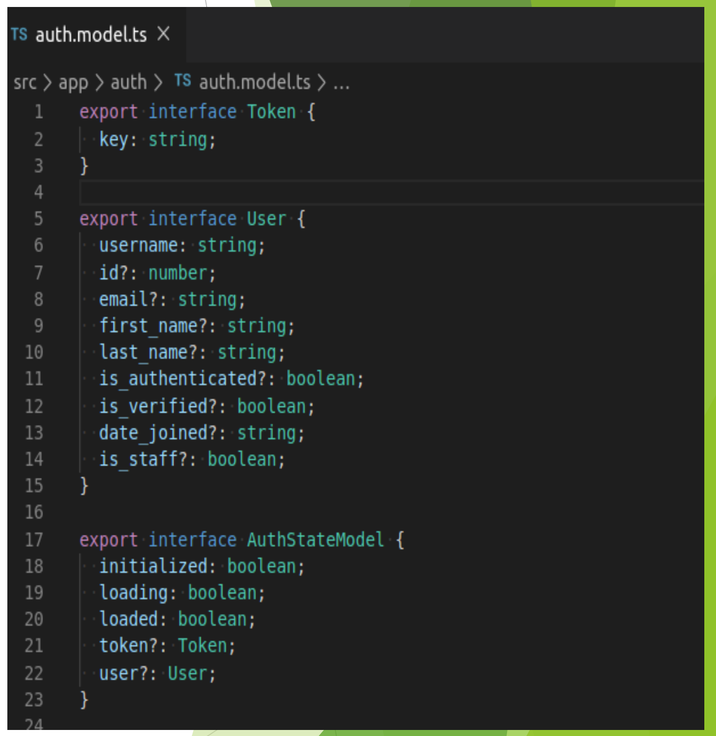
Після отримання даних зі сховища завдяки потоковому виведенню, шаблон компоненту може відображати дані за допомогою раніше сформованої підписки на селектор даних.

РОЗРОБКА МЕТОДІВ РЕАКТИВНОГО ВИВЕДЕННЯ ДАНИХ

Для створення об’єкту AuthState використовується декоратор @State з типом, що заданий інтерфейсом AuthStateModel. Поле defaults визначає потачковий стан об’єкту.



AuthStateModel



Створення об’єкту Авторизації(AuthState) в сховищі Store

РОЗРОБКА МЕТОДІВ РЕАКТИВНОГО ВИВЕДЕННЯ ДАНИХ

Створення об’єкту AuthState, його методів та інтерфейсів реалізовано засобами NGXS та TypeScript. Метод селективного вибору даних реалізований за допопомогою декоратору @Selector.

```
@Selector()
static isAuthenticated(state: AuthStateModel): boolean {
  return state.user ? state.user.is_authenticated : false;
}

@Selector()
static isVerified(state: AuthStateModel): boolean {
  return state.user ? state.user.is_verified : false;
}

@Selector()
static isLoading(state: AuthStateModel): boolean {
  return state.loading;
}

@Selector()
static isLoaded(state: AuthStateModel): boolean {
  return state.loaded;
}
```

Реалізація методу селективного вибору даних для AuthState

РОЗРОБКА МЕТОДІВ РЕАКТИВНОГО ВИВЕДЕННЯ ДАНИХ

Створення підписки на селектор даних зі сховища даних Store відбувається в компоненті і реалізовується за допомогою RxJS. Цей підхід дозволяє реалізувати потокове виведення даних зі сховища з використанням селективного вибору, що є важливою частиною методу реактивного виведення даних.

```
export class LoginComponent extends NgxsSingleFormComponent implements OnInit, OnDestroy {
  @Select(AuthState.isLoading) loading$: Observable<boolean>;
  @Select(MessagesState.getBackendError) backendErrors$: Observable<any>;

  constructor(
    protected store: Store,
    protected fb: FormBuilder,
    protected cd: ChangeDetectorRef,
    ) {
    super(fb, cd, store);
  }

  ngOnInit() {
    super.ngOnInit();
    this.store.dispatch(new Init());
  }

  protected performSubmit() {
    this.store.dispatch(new LoginWithEmailAndPassword(this.form.value));
  }
```

Створення підписки на селектор даних об’єкту AuthState

РОЗРОБКА МЕТОДІВ РЕАКТИВНОГО ВИВЕДЕННЯ ДАНИХ

Завдяки створенню підписки на селектор даних шаблон компоненту має доступ до потоку даних зі сховища, який надходить за допомогою селекторів, які в свою чергу відображають автоматично відслідковані дані з серверу.

```
uth.model.ts    login.component.html ×
app > auth > login > login.component.html > div.content
<div class="content" [@loadContent]="'open'">
  <div class="text-center">
    <div class="text-center">
      If you have not created an account yet, then please
      <a routerLink="/auth/register/"><strong>Sign Up</strong></a> first.
    </div>
  </div>
  <div id="content_inner">
    <div class="row">
      <div class="col-md-6 col-md-offset-3 login_form">
        <form [formGroup]="form" class="well login" (ngSubmit)="performSubmit()">
          <h2>Log In</h2>
          <div class="form-group">
            <rt-input-block class="email" name='email' required='false' [label]="formLabels['email']" [form]="form"
              [errors]="formErrors"></rt-input-block>
          </div>
          <div class="form-group">
            <rt-input-block class="password" type="password" name='password' [label]="formLabels['password']" [form]="form"
              [errors]="formErrors"></rt-input-block>
          </div>
          <input type="hidden" name="login-redirect_url" id="id_login-redirect_url">
          <p><a class="forgotPassword" routerLink='/auth/password-reset'>I've forgotten my password</a></p>
          <button mat-flat-button color="primary" type="submit" [disabled]="loading$ | async">Log In</button>
        </form>
      </div>
    </div>
  </div>
</div>
```

В даній реалізації реактивне виведення даних використовується для зміни властивості disabled кнопки Log In, відслідковуючи зміни сховища даних за допомогою підписки loading$ з компоненту та використання async pipe, що забезпечує виведення асинхронних даних в шаблоні.

Потокове реактивне виведення в шаблоні компоненту

РОЗРОБКА МЕТОДІВ РЕАКТИВНОГО ВИВЕДЕННЯ ДАНИХ

Після заповнення форми для входу і натисненні кнопки LOGIN в компоненті запускається функція performSubmit(), яка викликає керуючий сигнал і запускає в Store нову подію з AuthState. Подія робить запит до серверу через сервіс Auth і створює підписку на відповідь з серверу. Після отримання відповіді з серверу обробляє її і модифікує сховище даних

```
}

@Action(LoginWithEmailAndPassword)
loginWithEmailAndPassword(ctx: StateContext<AuthStateModel>, action: LoginWithEmailAndPassword) {
  ctx.patchState({loading: true, loaded: false});
  this.auth.signInWithEmailAndPassword(action.data)
    .subscribe(
      token => ctx.dispatch(new LoginSuccess(token)),
      err => ctx.dispatch(new LoginFail(err)),
    );
}
@Action(LoginSuccess)
setUserStateOnSuccess(ctx: StateContext<AuthStateModel>, event: LoginSuccess) {
  ctx.patchState({
    token: event.token,
  });
}
@Action(LoginFail)
loginFail({patchState, dispatch}: StateContext<AuthStateModel>, {err}: LoginFail) {
  patchState({loading: false, loaded: false});
  dispatch(new BackendError(err));
}

@Action(LoginRedirect)
loginRedirect(ctx: StateContext<AuthStateModel>) {
  ctx.dispatch(new Navigate([this.settings.LOGIN_URL]));
}

@Action([LoginSuccess, SendFormSuccess])
loginSuccess(ctx: StateContext<AuthStateModel>) {
  ctx.dispatch(new CheckSession());
  const redirect = this.route.snapshot.queryParams.redirect || this.settings.LOGIN_REDIRECT_URL;
  ctx.patchState({loading: false, loaded: true});
  ctx.dispatch(new Navigate([redirect]));
  this.scroll.toId('top');
}
```

```
export class LoginComponent extends NgxsSingleFormComponent implements OnInit, OnDestroy {
  @Select(AuthState.isLoading) loading$: Observable<boolean>;
  @Select(MessagesState.getBackendError) backendErrors$: Observable<any>;

  constructor(
    protected store: Store,
    protected fb: FormBuilder,
    protected cd: ChangeDetectorRef,
    ) {
    super(fb, cd, store);
  }

  ngOnInit() {
    super.ngOnInit();
    this.store.dispatch(new Init());
  }

  protected performSubmit() {
    this.store.dispatch(new LoginWithEmailAndPassword(this.form.value));
  }
```

Робота з подіями для об’єкту AuthState

РОЗРОБКА МЕТОДІВ РЕАКТИВНОГО ВИВЕДЕННЯ ДАНИХ

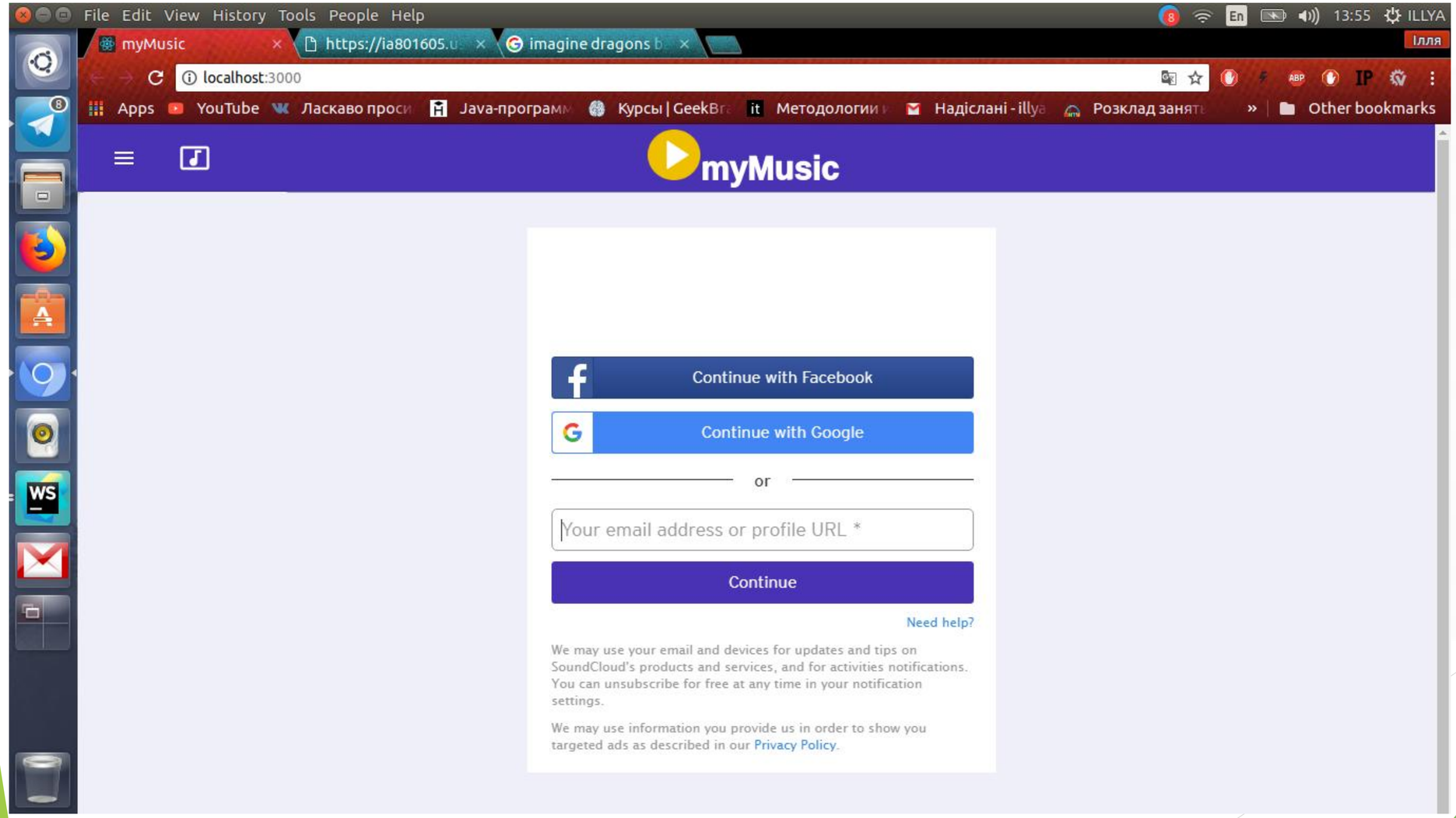
AuthService має в собі методи, які взаємодіють з сервером, створюючи потік даних через Observable, відповідь з сервера до сховища даних надходить через потік даних.

```
@Injectable()
export class AuthService {
  API_ENDPOINT = '/api/rest-auth';

  constructor(private http: HttpClient) { }

  signInWithEmailAndPassword(formValue): Observable<Token> {
    return this.http.post<Token>(`${this.API_ENDPOINT}/login/`, formValue);
  }

  registration(formValue): Observable<Token> {
      return this.http.post<Token>(`${this.API_ENDPOINT}/registration/`, formValue);
  }

  logout(): Observable<SimpleResponse> {
    return this.http.post<SimpleResponse>(`${this.API_ENDPOINT}/logout/`, {});
  }

  passwordReset(email: string): Observable<SimpleResponse> {
    return this.http.post<SimpleResponse>(`${this.API_ENDPOINT}/password/reset/`, email);
  }
```

Використання сервісів для роботи з сервором

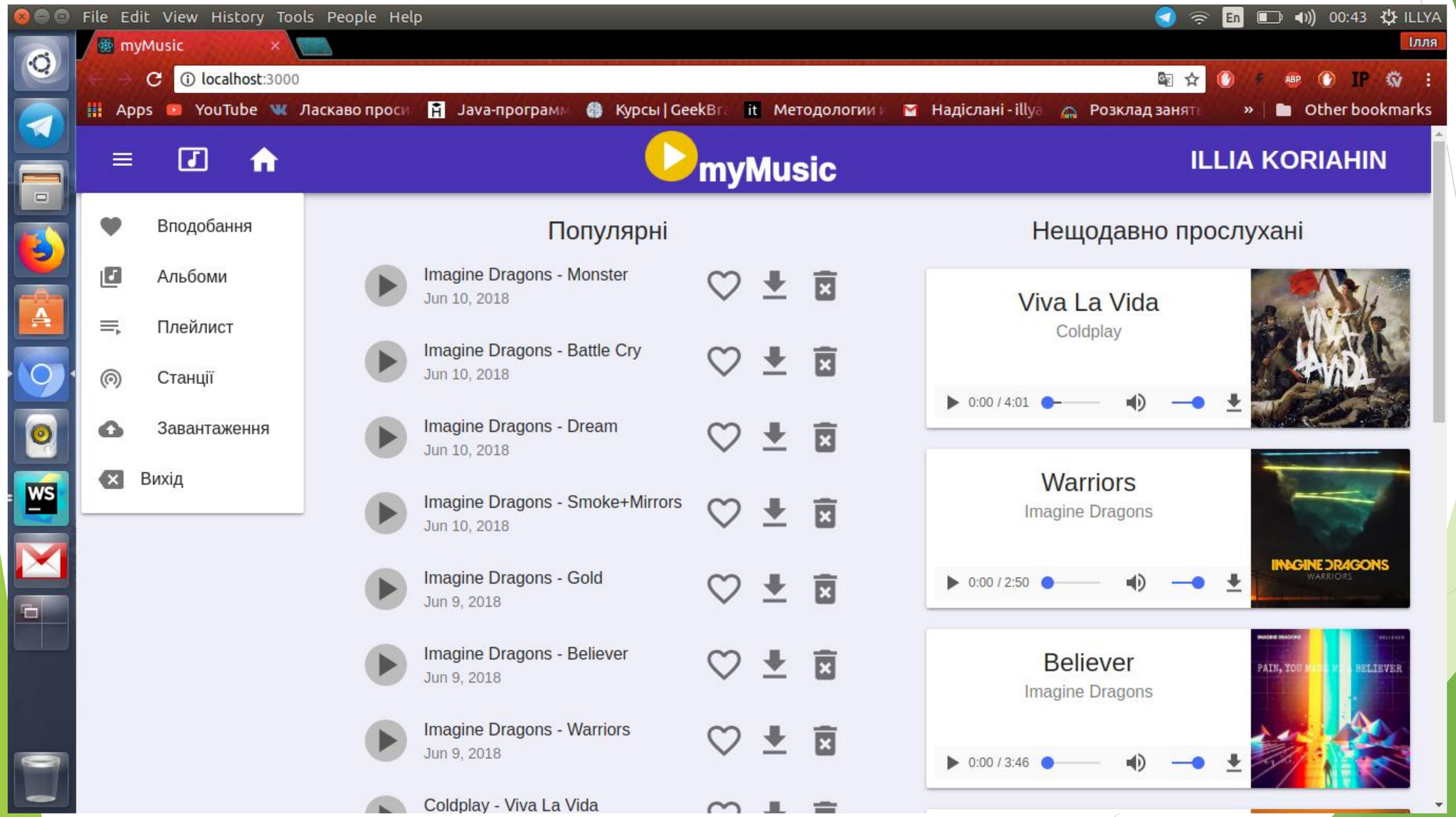
Тестування роботи додатку

Вхід в веб-сервіс «myMusic»



Тестування роботи додатку

Головна сторінка після авторизації



Результати роботи та висновки

Розроблено та вирішено такі задачі:

- Розробка проміжного сховища даних Store.
- Надання можливості збереження багаторівневих структур даних.
- Підтримка різних типів даних.
- Підтримка потокового виведення даних.
- Надання можливості селективного вибору даних.
- Автоматичне відслідковування даних.
- Можливість обробки відповіді з серверу.
- Наявність початкового стану сховище даних.
- Швидкий доступ до сховище з будь-якого компоненту

Дякую за Увагу!