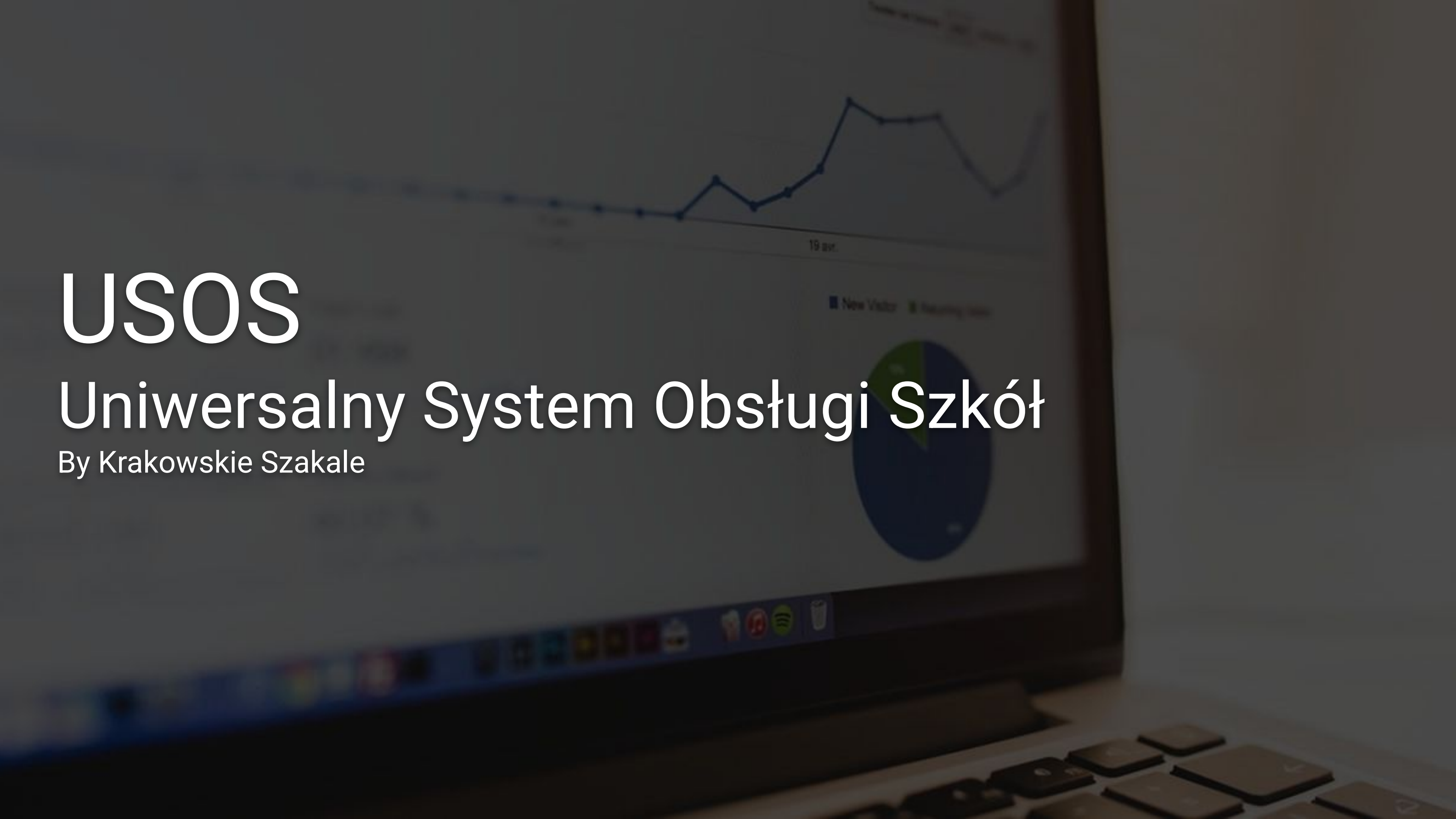


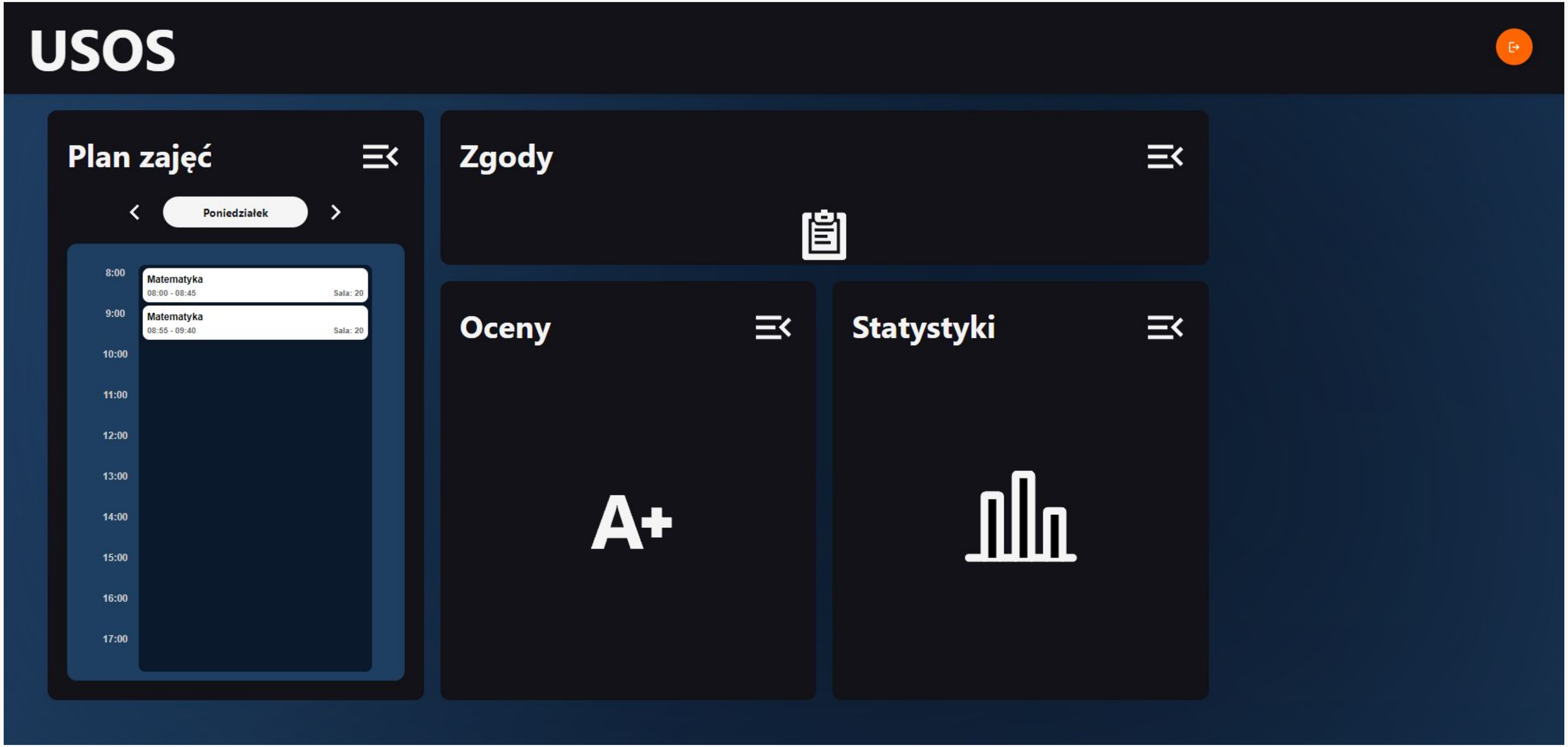
USOS

Uniwersalny System Obsługi Szkół

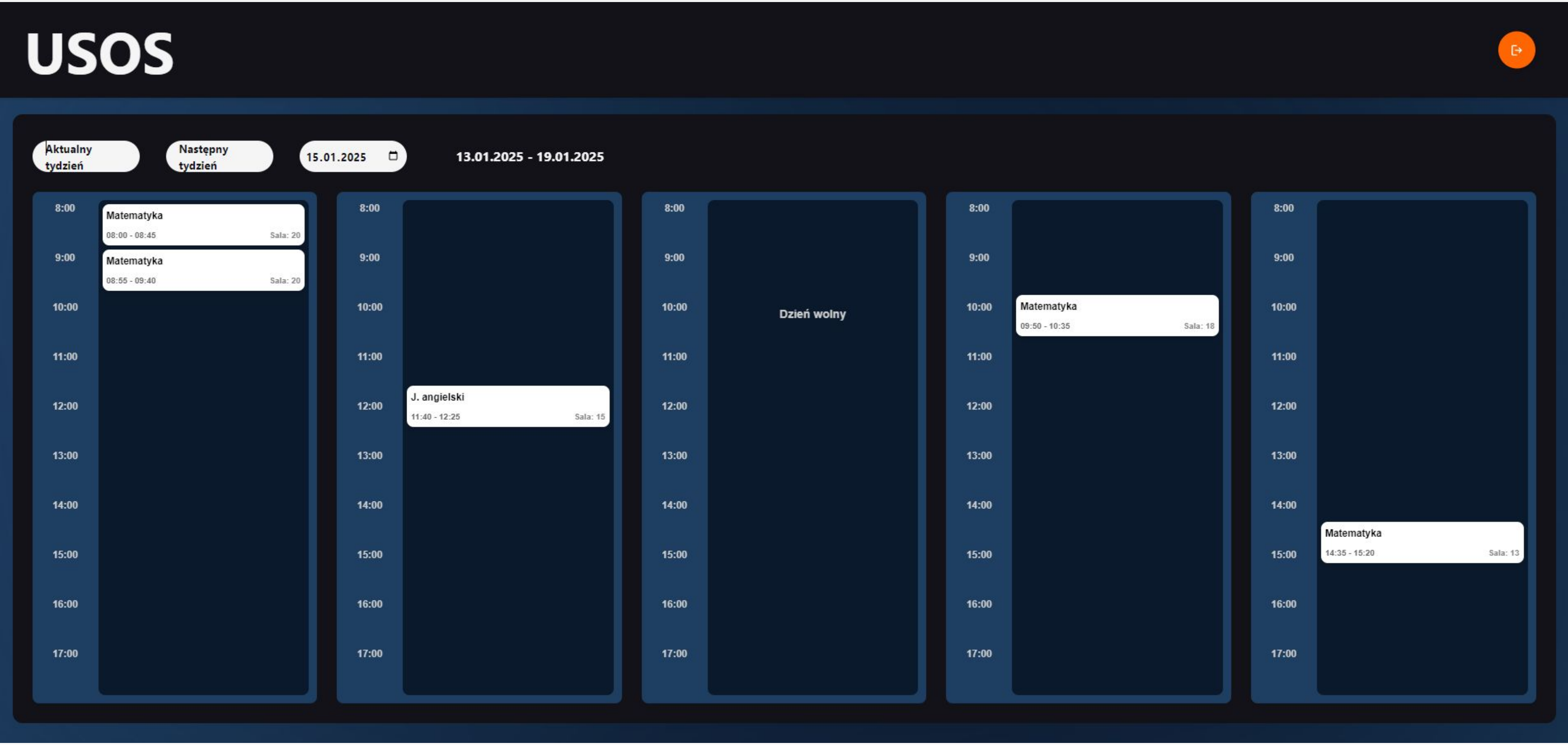
By Krakowskie Szakale

Interfejs użytkownika, UI/UX, Podsumowanie projektu

Nauczyciel - główny widok



Nauczyciel - plan zajęć



USOS

Aktualny tydzień

Następny tydzień

15.01.2025

13.01.2025 - 19.01.2025

Matematyka
08:00 - 08:45 Sala: 20

Matematyka
08:55 - 09:40 Sala: 20

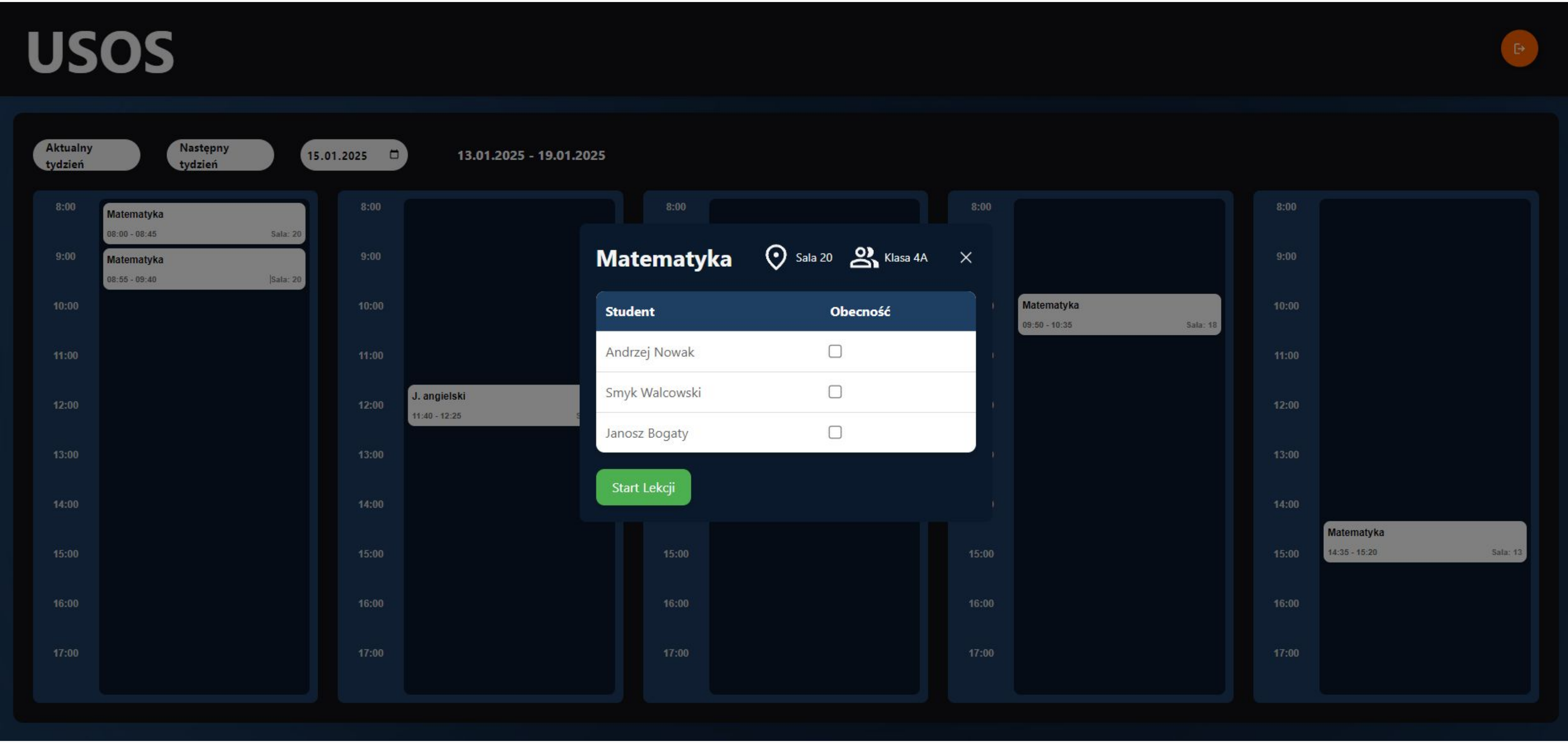
J. angielski
11:40 - 12:25 Sala: 15

Dzień wolny

Matematyka
09:50 - 10:35 Sala: 18

Matematyka
14:35 - 15:20 Sala: 13

Coś z ui



USOS

Aktualny tydzień | Następny tydzień | 15.01.2025 | 13.01.2025 - 19.01.2025

Matematyka 08:00 - 08:45 Sala: 20

Matematyka 08:55 - 09:40 Sala: 20

J. angielski 11:40 - 12:25

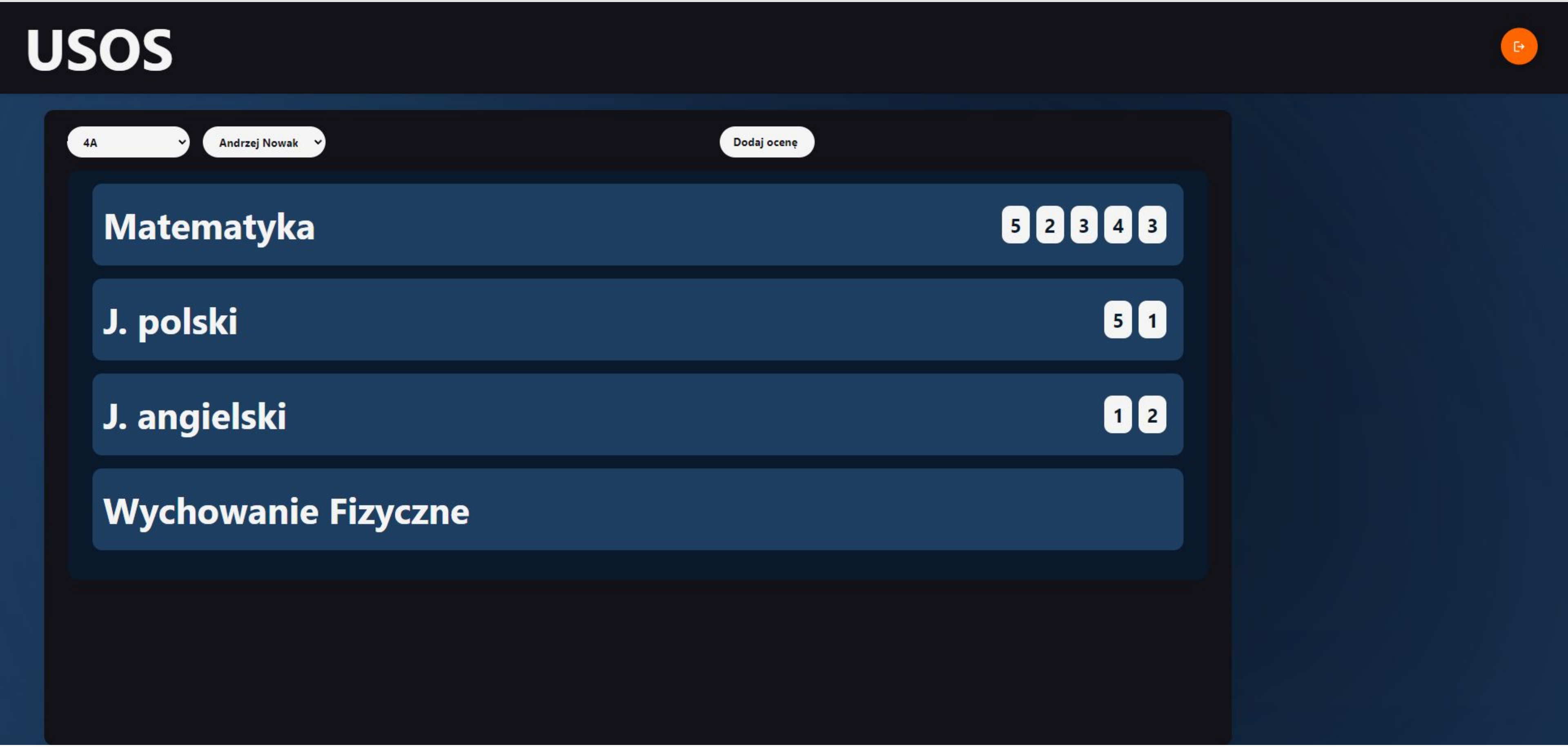
Matematyka 09:50 - 10:35 Sala: 18

Matematyka 14:35 - 15:20 Sala: 13

Matematyka Sala 20 Klasa 4A

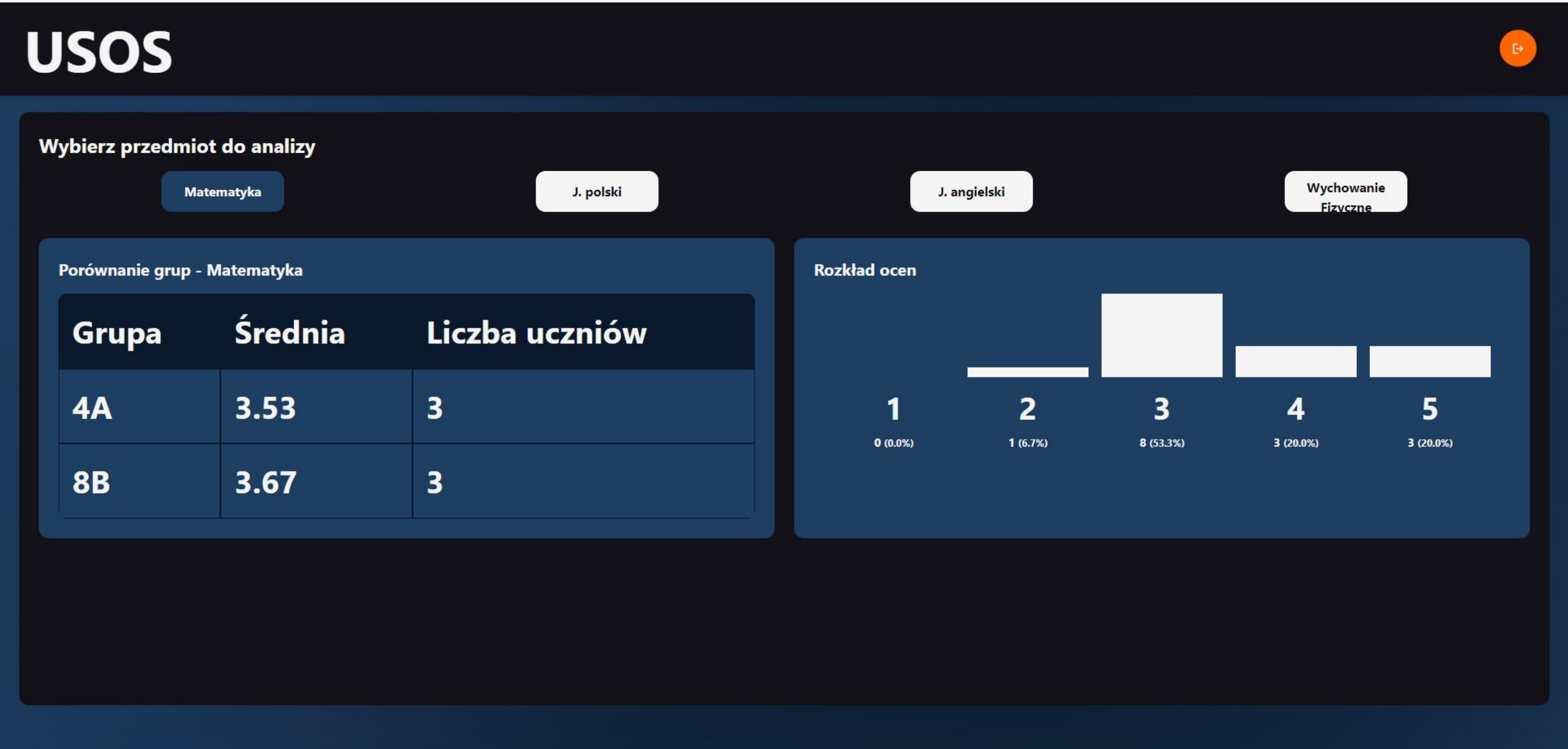
Student	Obecność
Andrzej Nowak	☐
Smyk Walcowski	☐
Janosz Bogaty	☐

Nauczyciel - Wyświetlanie ocen



Matematyka	5 2 3 4 3
J. polski	5 1
J. angielski	1 2
Wychowanie Fizyczne	

Statystyki u nauczyciela



USOS

Wybierz przedmiot do analizy

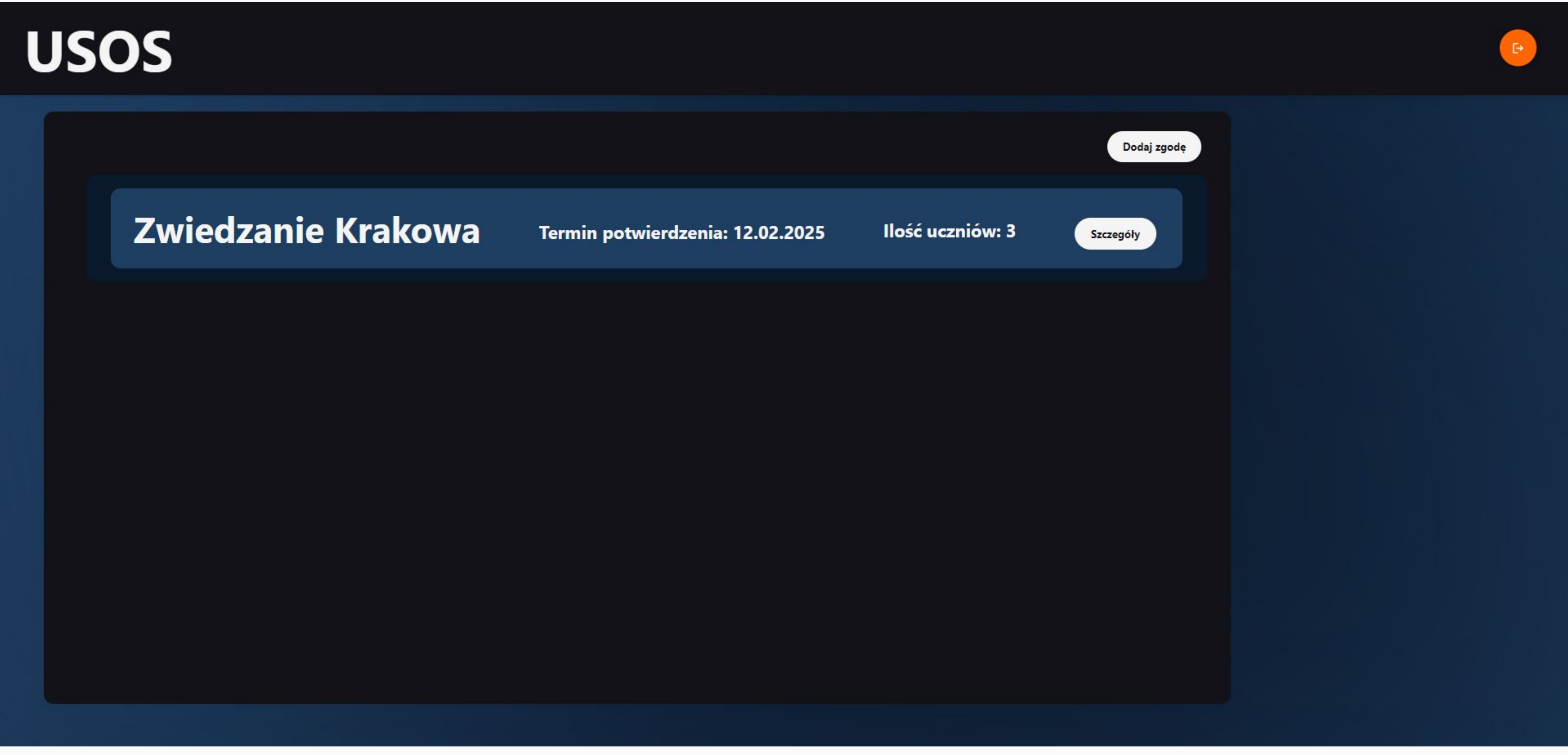
Matematyka | J. polski | J. angielski | Wychowanie Fizyczne

Porównanie grup - Matematyka

Grupa	Średnia	Liczba uczniów
4A	3.53	3
8B	3.67	3

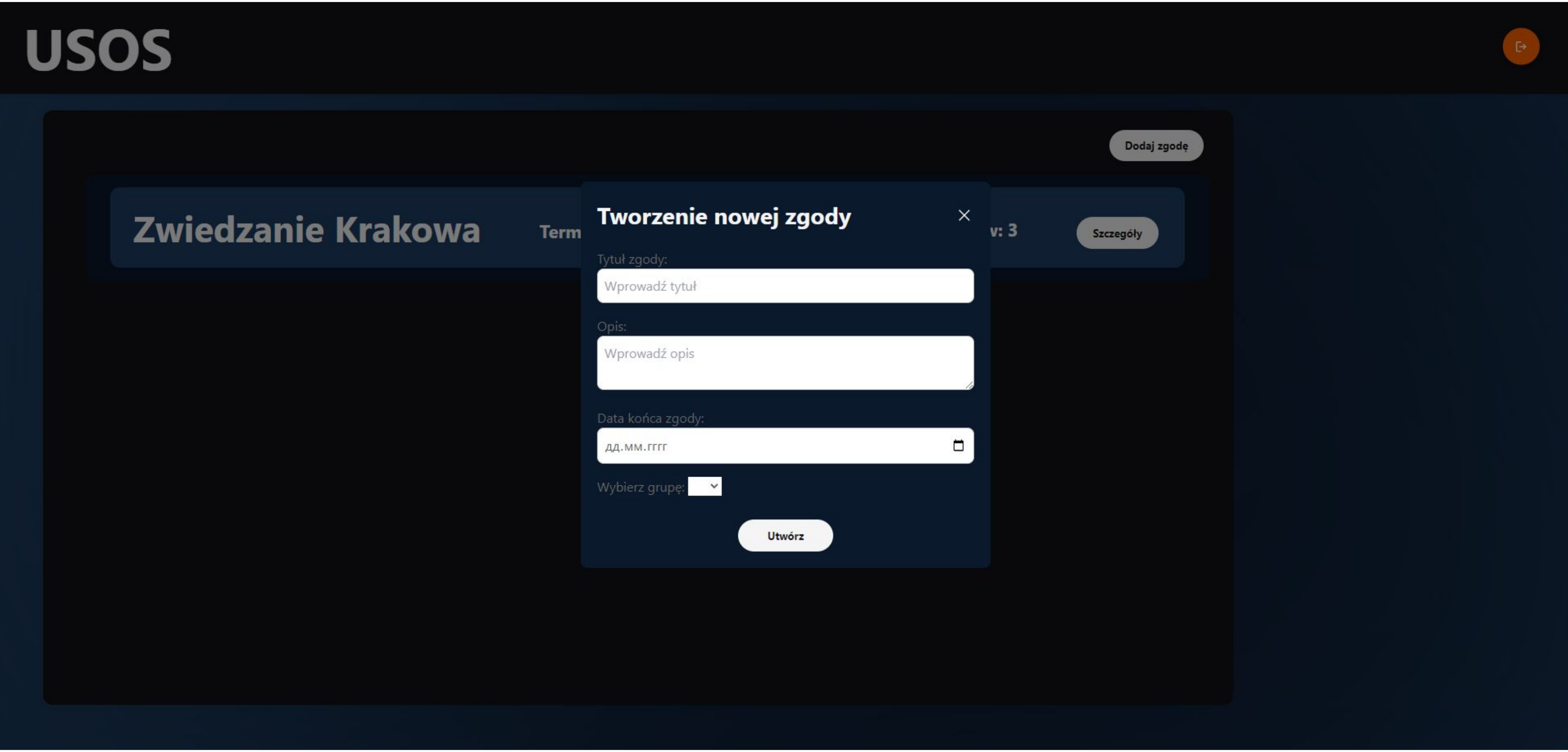
Rozkład ocen

Nauczyciel - Wyświetlanie utworzonych zgód



Zwiedzanie Krakowa Termin potwierdzenia: 12.02.2025 Ilość uczniów: 3

Nauczyciel - Tworzenie zgody elektronicznej


USOS

Dodaj zgodę

Zwiedzanie Krakowa

Term

v: 3

Szczegóły

Tworzenie nowej zgody

Tytuł zgody:

Wprowadź tytuł

Opis:

Wprowadź opis

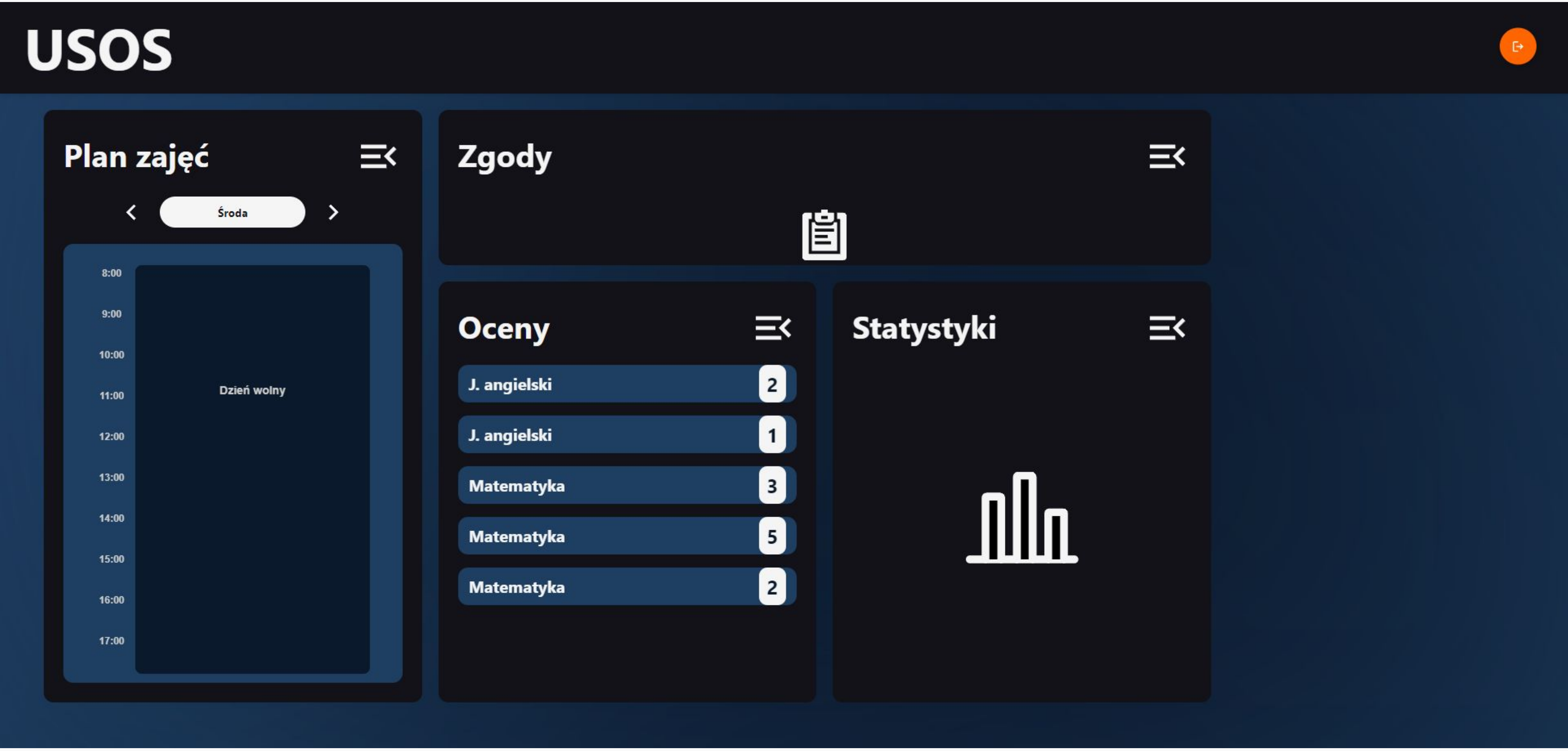
Data końca zgody:

дд.мм.гггг

Wybierz grupę:

Utwórz

Rodzic - Widok główny



USOS

Plan zajęć

Środa

8:00
9:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00

Dzień wolny

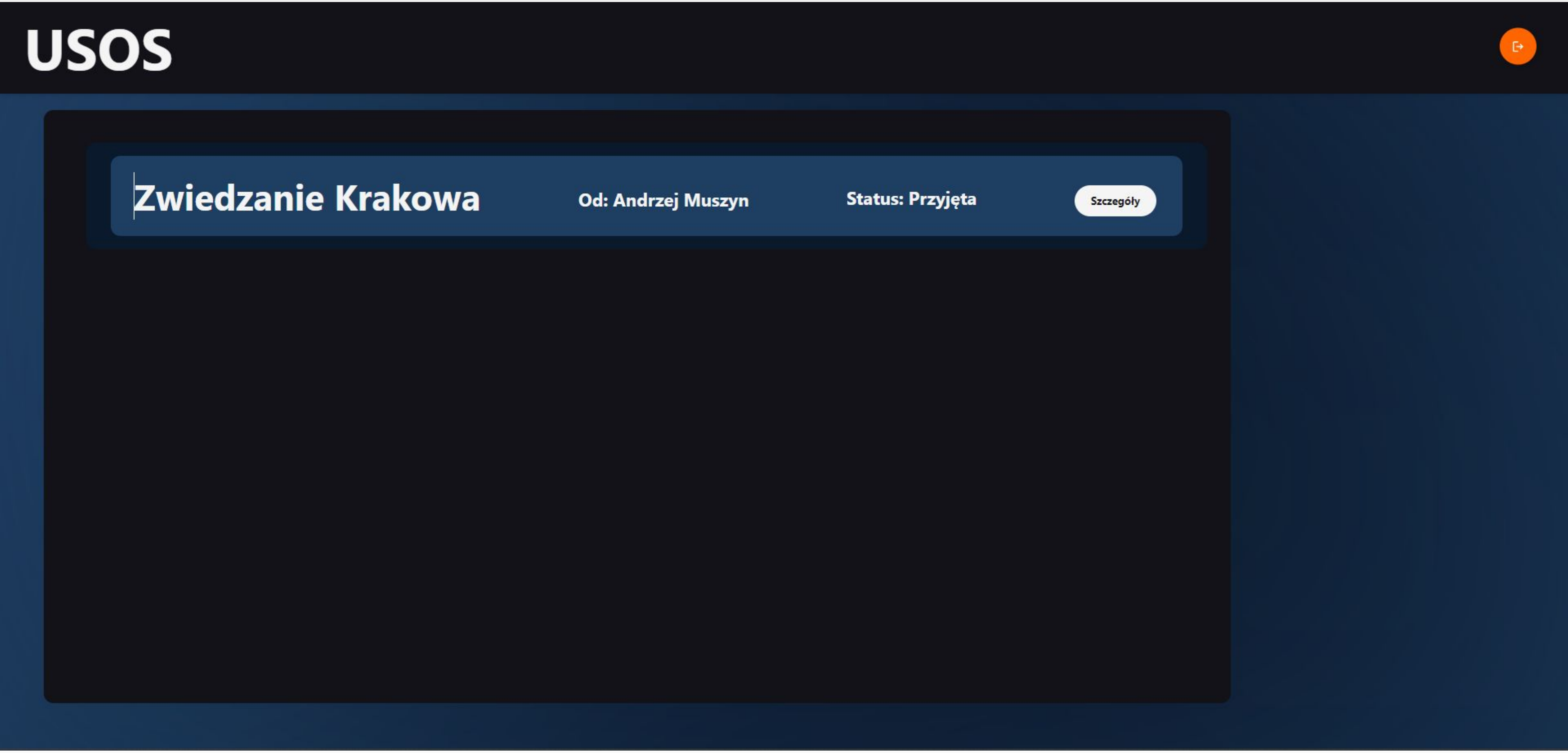
Zgody

Oceny

J. angielski	2
J. angielski	1
Matematyka	3
Matematyka	5
Matematyka	2

Statystyki

Rodzic - Wyświetlanie zgód



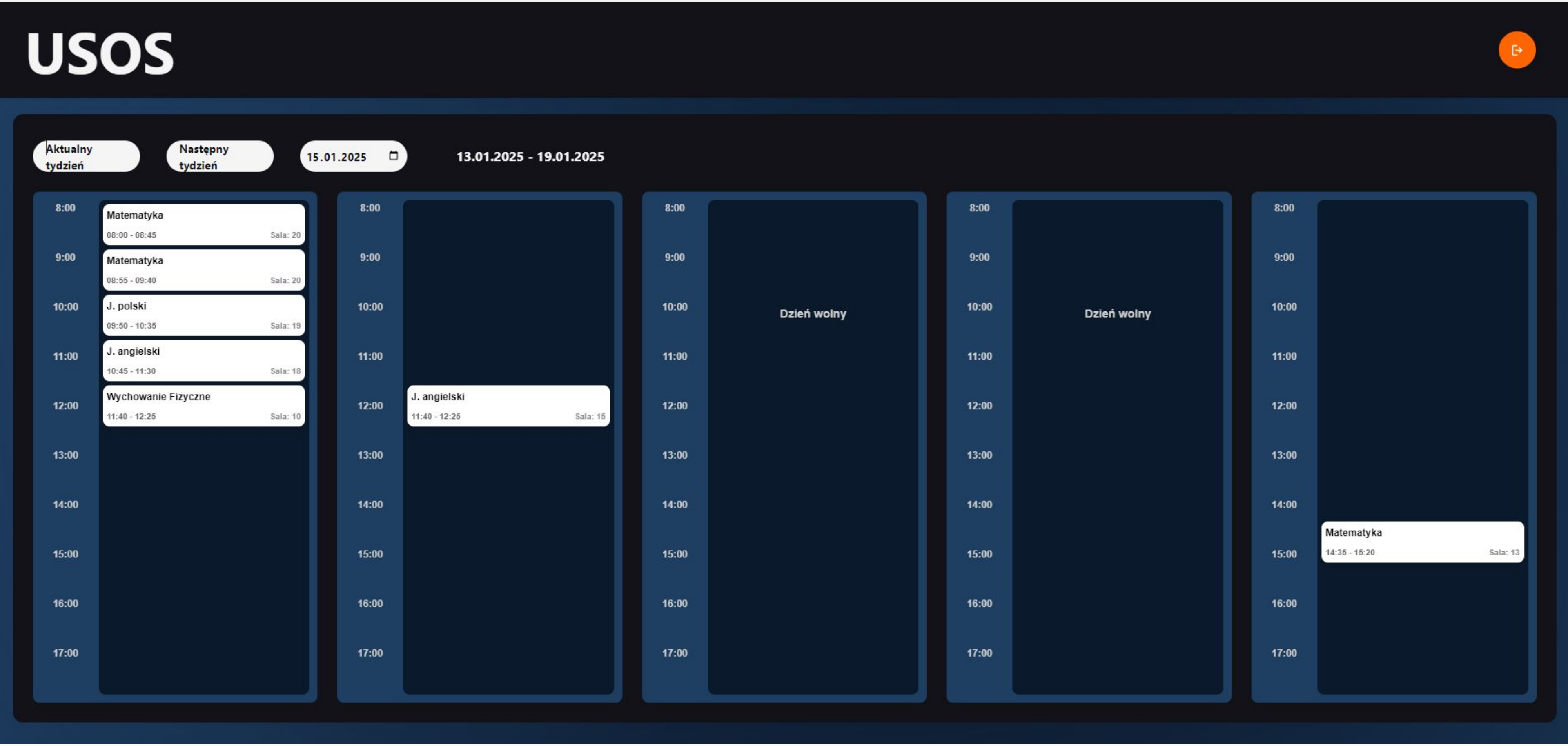
USOS

Zwiedzanie Krakowa

Od: Andrzej Muszyn

Status: Przyjęta

Rodzic - Plan lekcji



USOS

Aktualny tydzień | Następny tydzień | 15.01.2025 | 13.01.2025 - 19.01.2025

Matematyka
08:00 - 08:45 Sala: 20

Matematyka
08:55 - 09:40 Sala: 20

J. polski
09:50 - 10:35 Sala: 19

J. angielski
10:45 - 11:30 Sala: 18

Wychowanie Fizyczne
11:40 - 12:25 Sala: 10

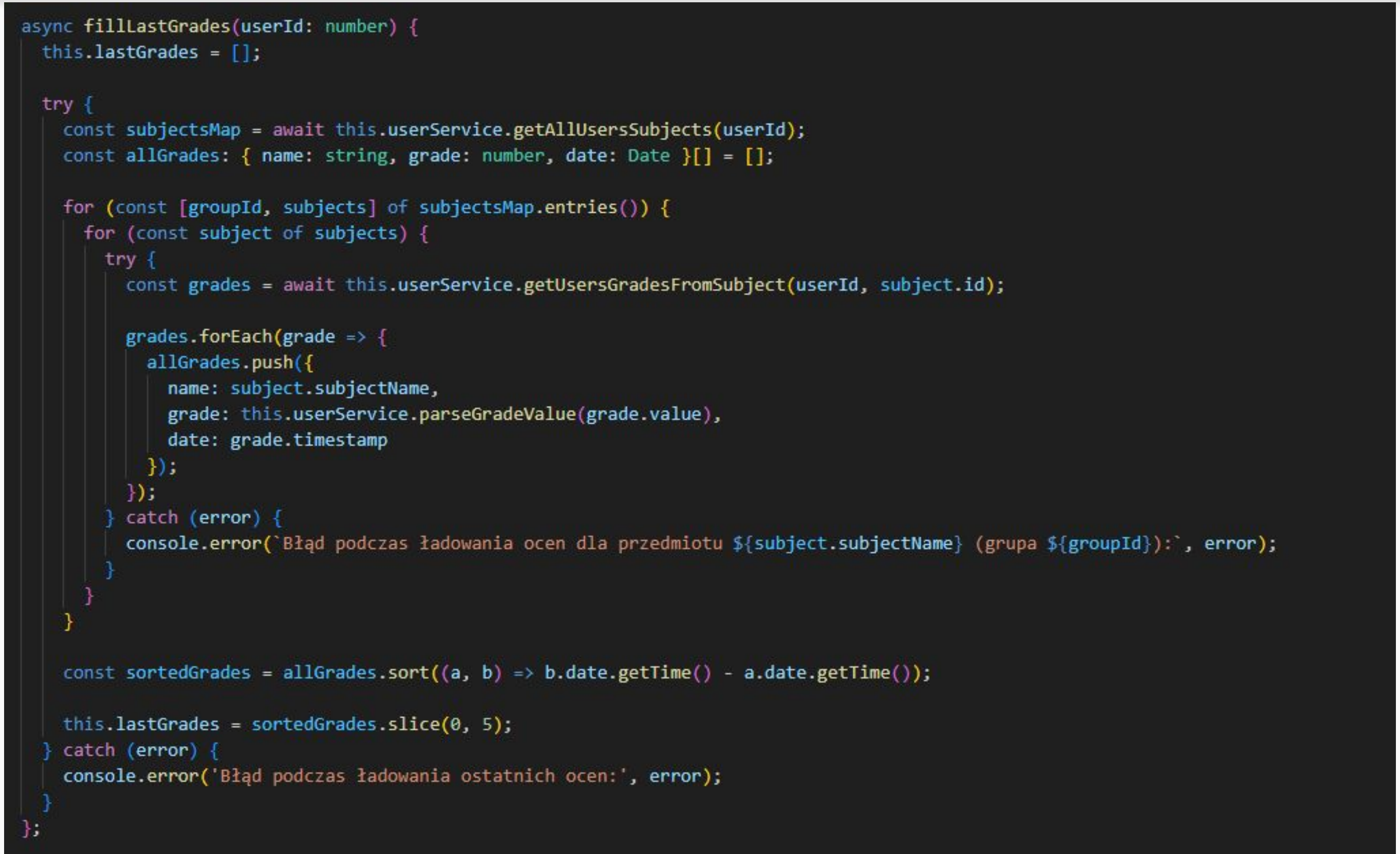
J. angielski
11:40 - 12:25 Sala: 15

Dzień wolny

Dzień wolny

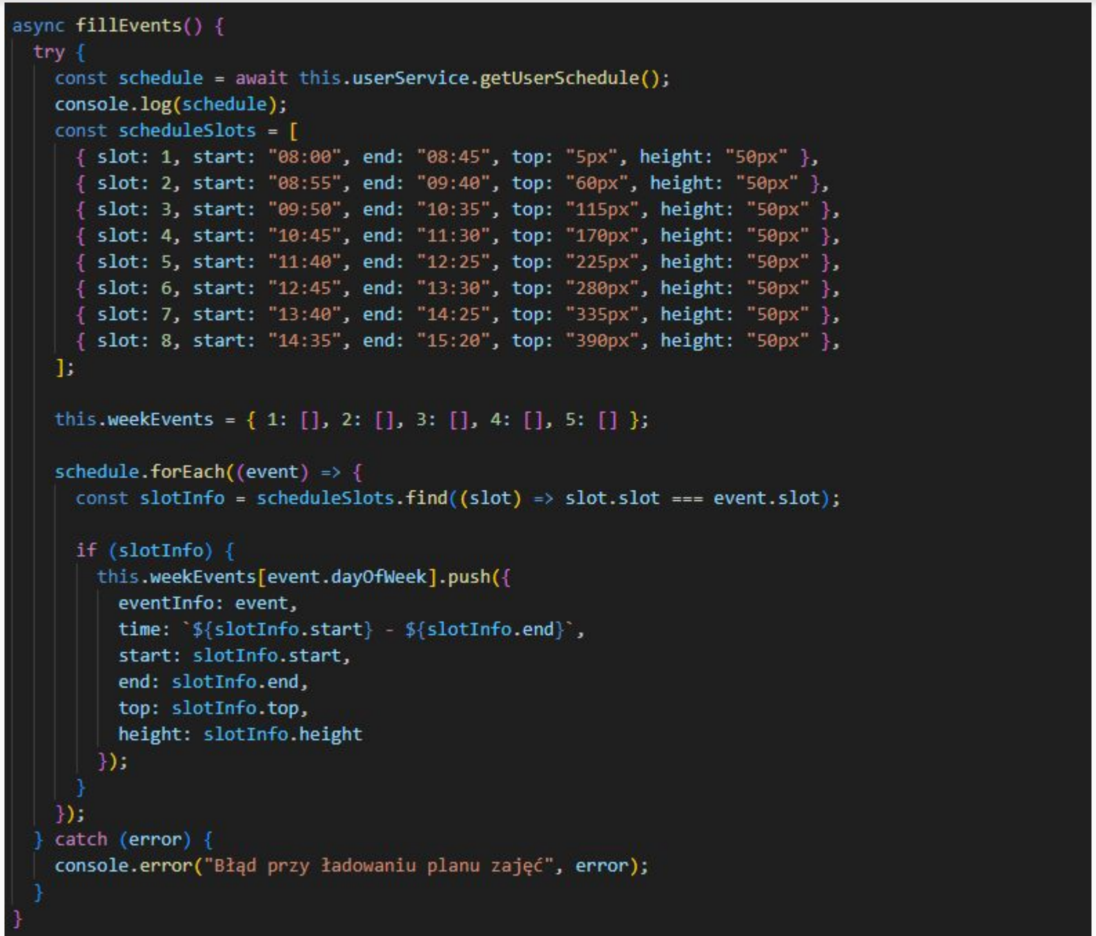
Matematyka
14:35 - 15:20 Sala: 13

Przykładowy kod po stronie UI - fillLastGrades



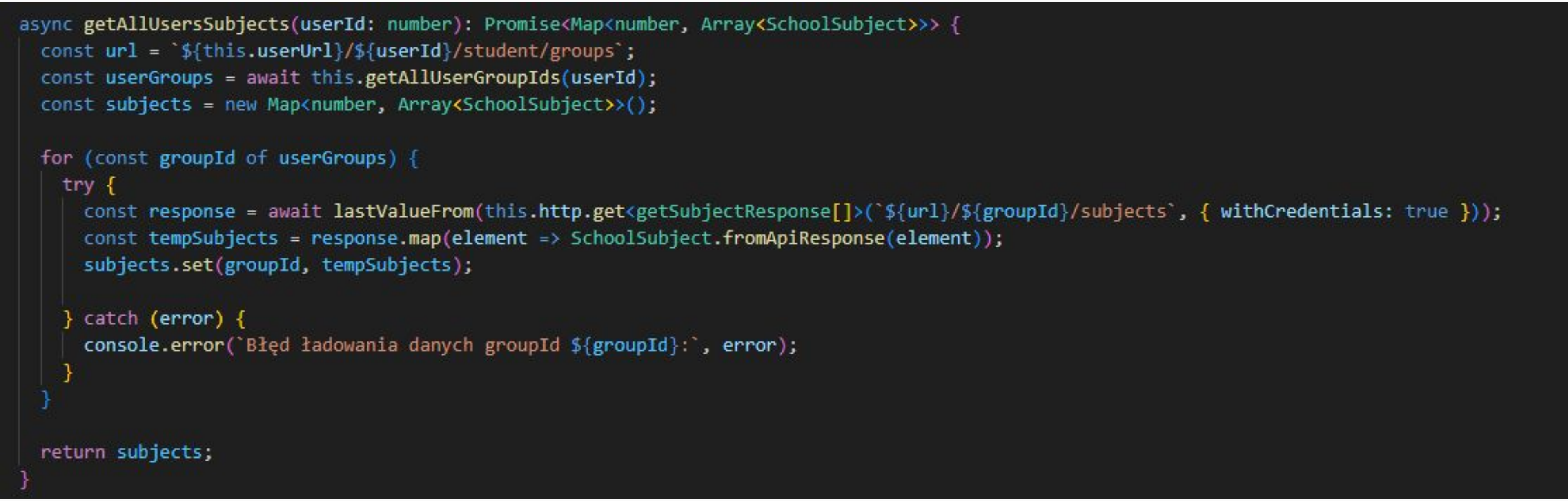
```
async fillLastGrades(userId: number) {
  this.lastGrades = [];

  try {
    const subjectsMap = await this.userService.getAllUsersSubjects(userId);
    const allGrades: { name: string, grade: number, date: Date }[] = [];

    for (const [groupId, subjects] of subjectsMap.entries()) {
      for (const subject of subjects) {
        try {
          const grades = await this.userService.getUsersGradesFromSubject(userId, subject.id);

          grades.forEach(grade => {
            allGrades.push({
              name: subject.subjectName,
              grade: this.userService.parseGradeValue(grade.value),
              date: grade.timestamp
            });
          });
        } catch (error) {
          console.error(`Błąd podczas ładowania ocen dla przedmiotu ${subject.subjectName} (grupa ${groupId}):`, error);
        }
      }
    }

    const sortedGrades = allGrades.sort((a, b) => b.date.getTime() - a.date.getTime());

    this.lastGrades = sortedGrades.slice(0, 5);
  } catch (error) {
    console.error('Błąd podczas ładowania ostatnich ocen:', error);
  }
};
```

Przykładowy kod po stronie UI - fillEvents



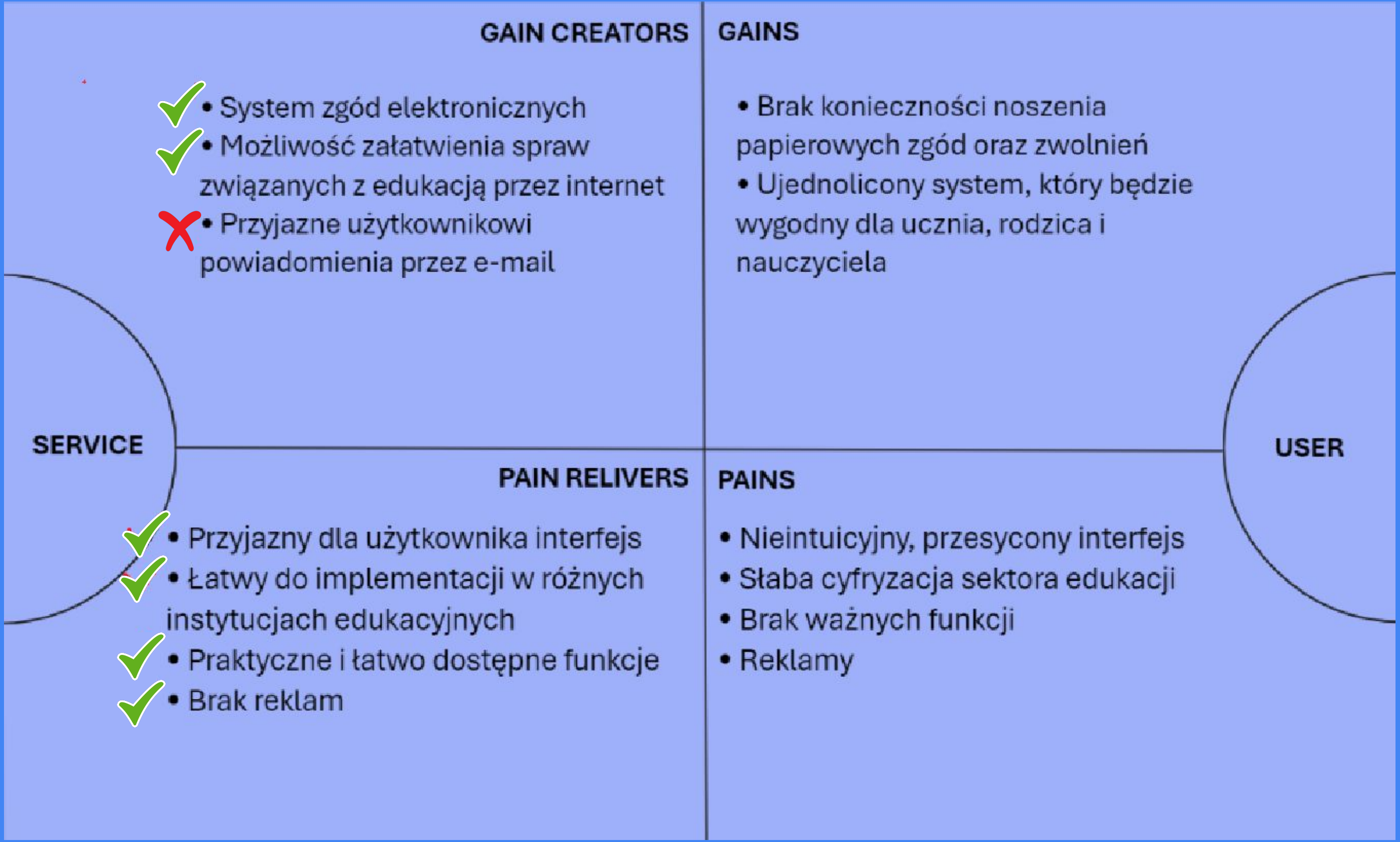
```
async fillEvents() {
  try {
    const schedule = await this.userService.getUserSchedule();
    console.log(schedule);
    const scheduleSlots = [
      { slot: 1, start: "08:00", end: "08:45", top: "5px", height: "50px" },
      { slot: 2, start: "08:55", end: "09:40", top: "60px", height: "50px" },
      { slot: 3, start: "09:50", end: "10:35", top: "115px", height: "50px" },
      { slot: 4, start: "10:45", end: "11:30", top: "170px", height: "50px" },
      { slot: 5, start: "11:40", end: "12:25", top: "225px", height: "50px" },
      { slot: 6, start: "12:45", end: "13:30", top: "280px", height: "50px" },
      { slot: 7, start: "13:40", end: "14:25", top: "335px", height: "50px" },
      { slot: 8, start: "14:35", end: "15:20", top: "390px", height: "50px" },
    ];

    this.weekEvents = { 1: [], 2: [], 3: [], 4: [], 5: [] };

    schedule.forEach((event) => {
      const slotInfo = scheduleSlots.find((slot) => slot.slot === event.slot);

      if (slotInfo) {
        this.weekEvents[event.dayOfWeek].push({
          eventInfo: event,
          time: `${slotInfo.start} - ${slotInfo.end}`,
          start: slotInfo.start,
          end: slotInfo.end,
          top: slotInfo.top,
          height: slotInfo.height
        });
      }
    });
  } catch (error) {
    console.error("Błąd przy ładowaniu planu zajęć", error);
  }
}
```

Przykładowy kod po stronie UI - getAllUsersSubjects



```
async getAllUsersSubjects(userId: number): Promise<Map<number, Array<SchoolSubject>>> {
  const url = `${this.userUrl}/${userId}/student/groups`;
  const userGroups = await this.getAllUserGroupIds(userId);
  const subjects = new Map<number, Array<SchoolSubject>>();

  for (const groupId of userGroups) {
    try {
      const response = await lastValueFrom(this.http.get<getSubjectResponse[]>(`${url}/${groupId}/subjects`, { withCredentials: true }));
      const tempSubjects = response.map(element => SchoolSubject.fromApiResponse(element));
      subjects.set(groupId, tempSubjects);

    } catch (error) {
      console.error(`Błęd ładowania danych groupId ${groupId}:`, error);
    }
  }

  return subjects;
}
```

Business value canvas



GAIN CREATORS

- ✓ System zgód elektronicznych
- ✓ Możliwość załatwienia spraw związanych z edukacją przez internet
- ✗ Przyjazne użytkownikowi powiadomienia przez e-mail

GAINS

- Brak konieczności noszenia papierowych zgód oraz zwolnień
- Ujednolicony system, który będzie wygodny dla ucznia, rodzica i nauczyciela

SERVICE

USER

PAIN RELIVERS

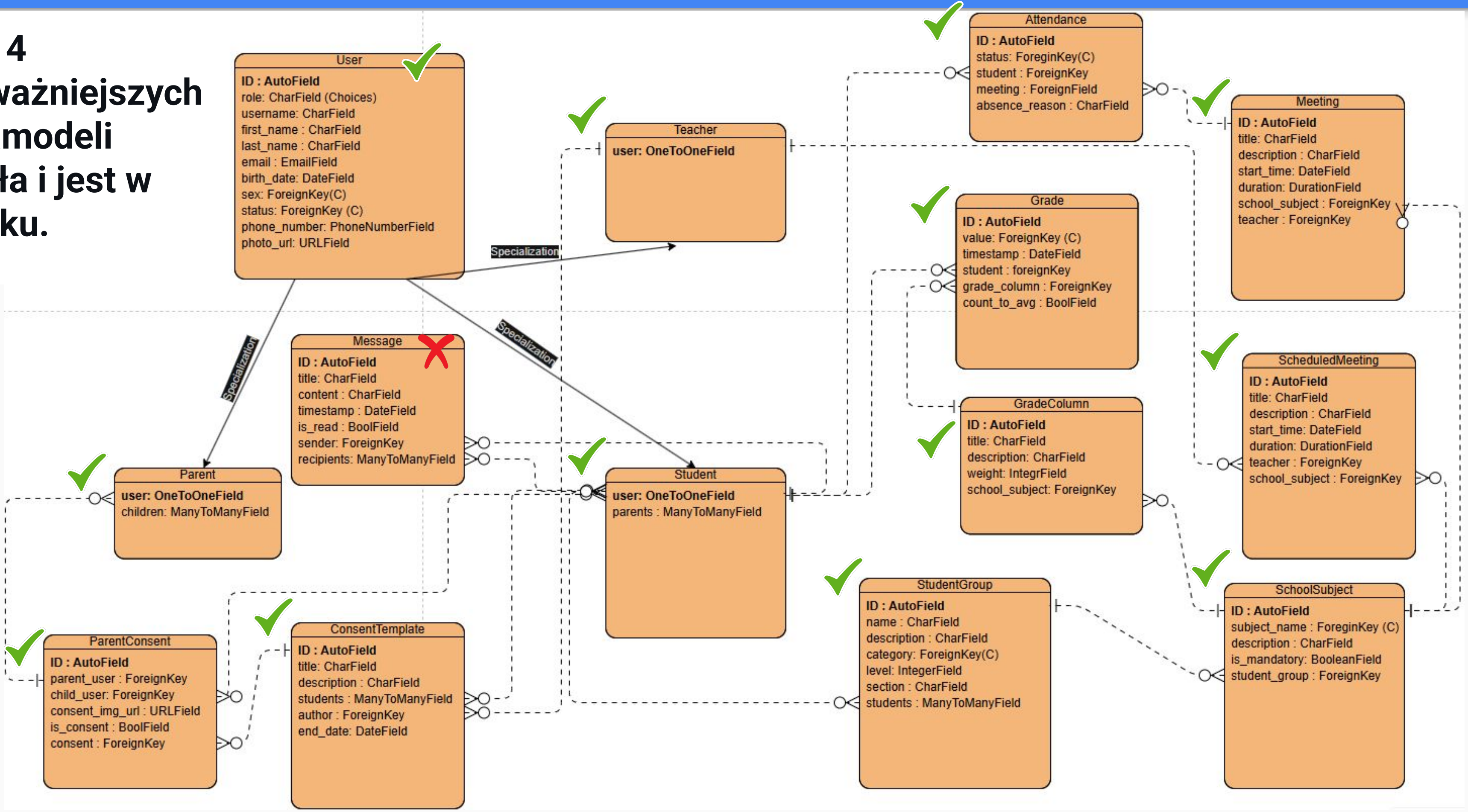
- ✓ Przyjazny dla użytkownika interfejs
- ✓ Łatwy do implementacji w różnych instytucjach edukacyjnych
- ✓ Praktyczne i łatwo dostępne funkcje
- ✓ Brak reklam

PAINS

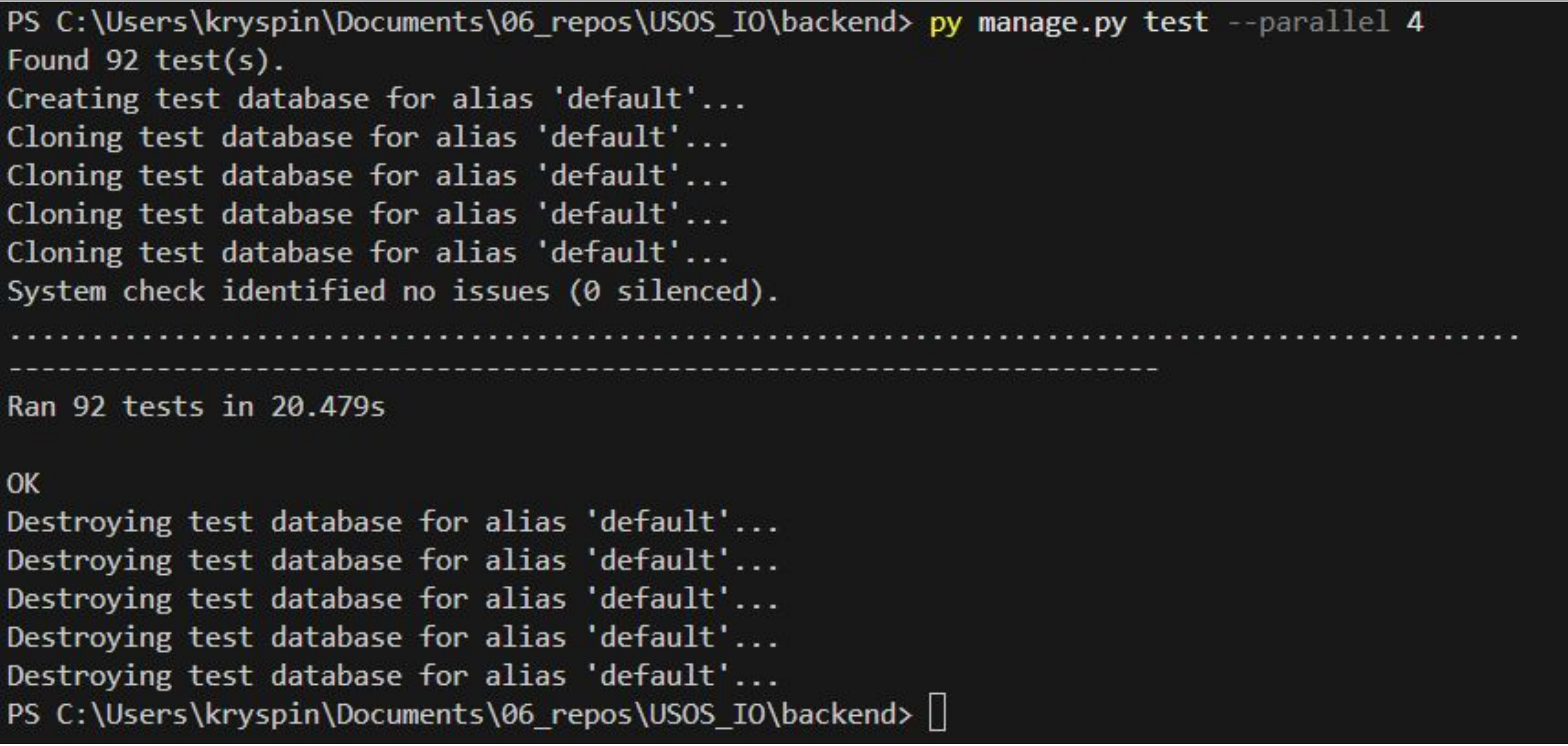
- Nieintuicyjny, przesycony interfejs
- Słaba cyfryzacja sektora edukacji
- Brak ważnych funkcji
- Reklamy

Data model diagram (ERD)

13/14 Najważniejszych datamodeli działa i jest w użytku.

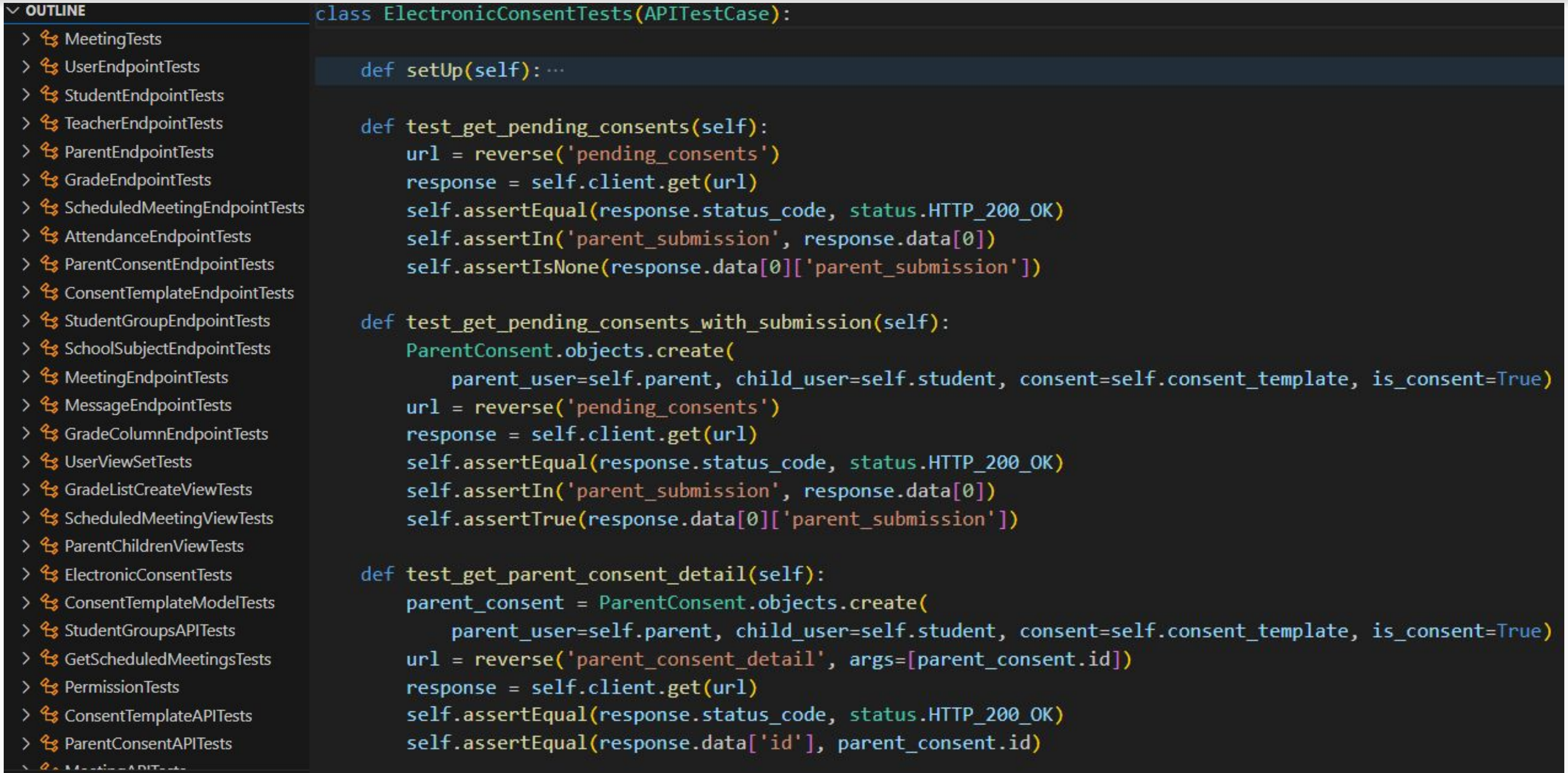


Testy



```
PS C:\Users\kryspin\Documents\06_repos\USOS_IO\backend> py manage.py test --parallel 4
Found 92 test(s).
Creating test database for alias 'default'...
Cloning test database for alias 'default'...
Cloning test database for alias 'default'...
Cloning test database for alias 'default'...
Cloning test database for alias 'default'...
System check identified no issues (0 silenced).
............................................................................................
----------------------------------------------------------------------
Ran 92 tests in 20.479s

OK
Destroying test database for alias 'default'...
Destroying test database for alias 'default'...
Destroying test database for alias 'default'...
Destroying test database for alias 'default'...
Destroying test database for alias 'default'...
PS C:\Users\kryspin\Documents\06_repos\USOS_IO\backend>
```

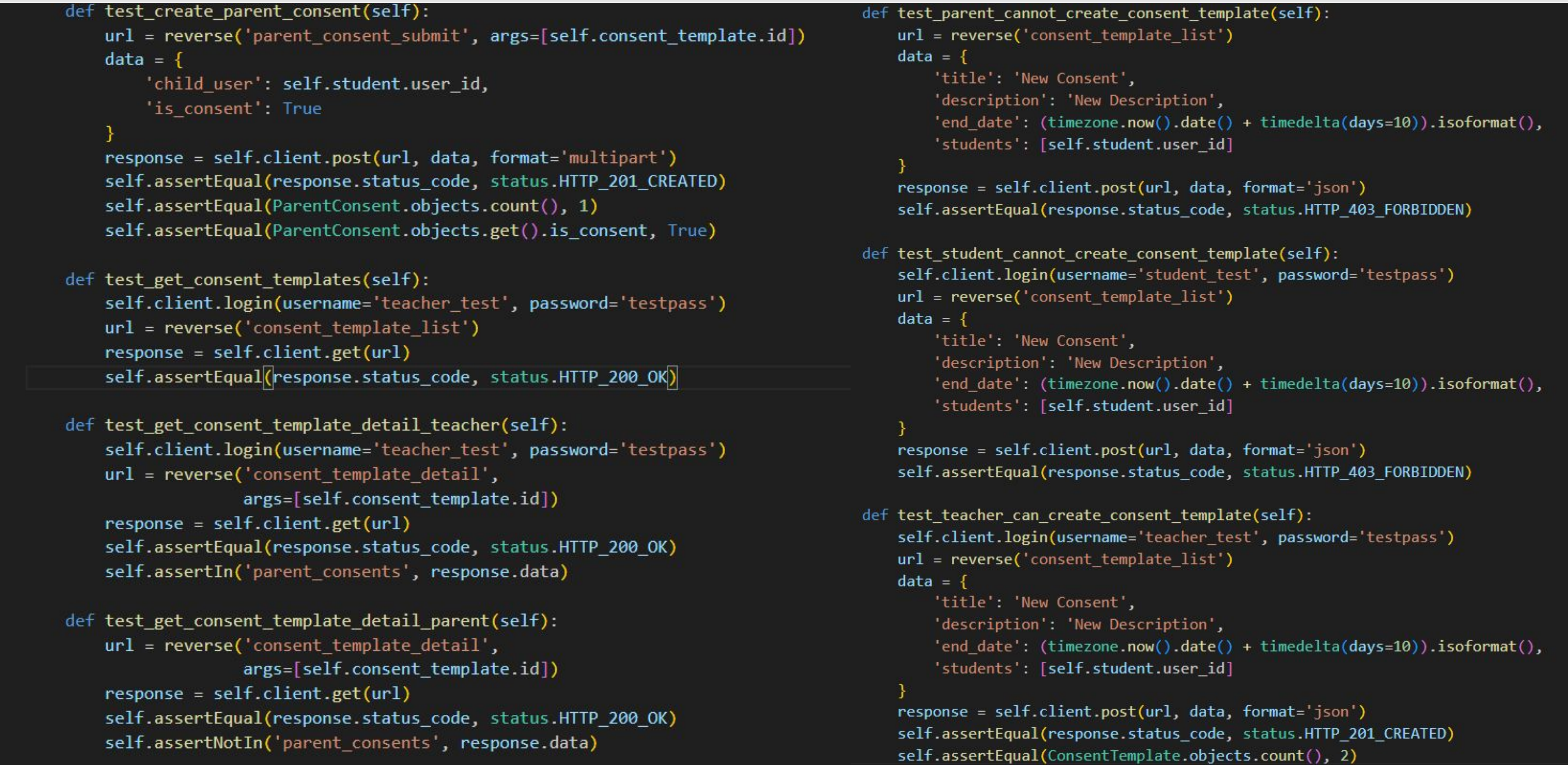
Przykładowe unittesty



OUTLINE
- MeetingTests
- UserEndpointTests
- StudentEndpointTests
- TeacherEndpointTests
- ParentEndpointTests
- GradeEndpointTests
- ScheduledMeetingEndpointTests
- AttendanceEndpointTests
- ParentConsentEndpointTests
- ConsentTemplateEndpointTests
- StudentGroupEndpointTests
- SchoolSubjectEndpointTests
- MeetingEndpointTests
- MessageEndpointTests
- GradeColumnEndpointTests
- UserViewSetTests
- GradeListCreateViewTests
- ScheduledMeetingViewTests
- ParentChildrenViewTests
- ElectronicConsentTests
- ConsentTemplateModelTests
- StudentGroupsAPITests
- GetScheduledMeetingsTests
- PermissionTests
- ConsentTemplateAPITests
- ParentConsentAPITests

```python
class ElectronicConsentTests(APITestCase):

    def setUp(self): ···

    def test_get_pending_consents(self):
        url = reverse('pending_consents')
        response = self.client.get(url)
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertIn('parent_submission', response.data[0])
        self.assertIsNone(response.data[0]['parent_submission'])

    def test_get_pending_consents_with_submission(self):
        ParentConsent.objects.create(
            parent_user=self.parent, child_user=self.student, consent=self.consent_template, is_consent=True)
        url = reverse('pending_consents')
        response = self.client.get(url)
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertIn('parent_submission', response.data[0])
        self.assertTrue(response.data[0]['parent_submission'])

    def test_get_parent_consent_detail(self):
        parent_consent = ParentConsent.objects.create(
            parent_user=self.parent, child_user=self.student, consent=self.consent_template, is_consent=True)
        url = reverse('parent_consent_detail', args=[parent_consent.id])
        response = self.client.get(url)
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertEqual(response.data['id'], parent_consent.id)
```

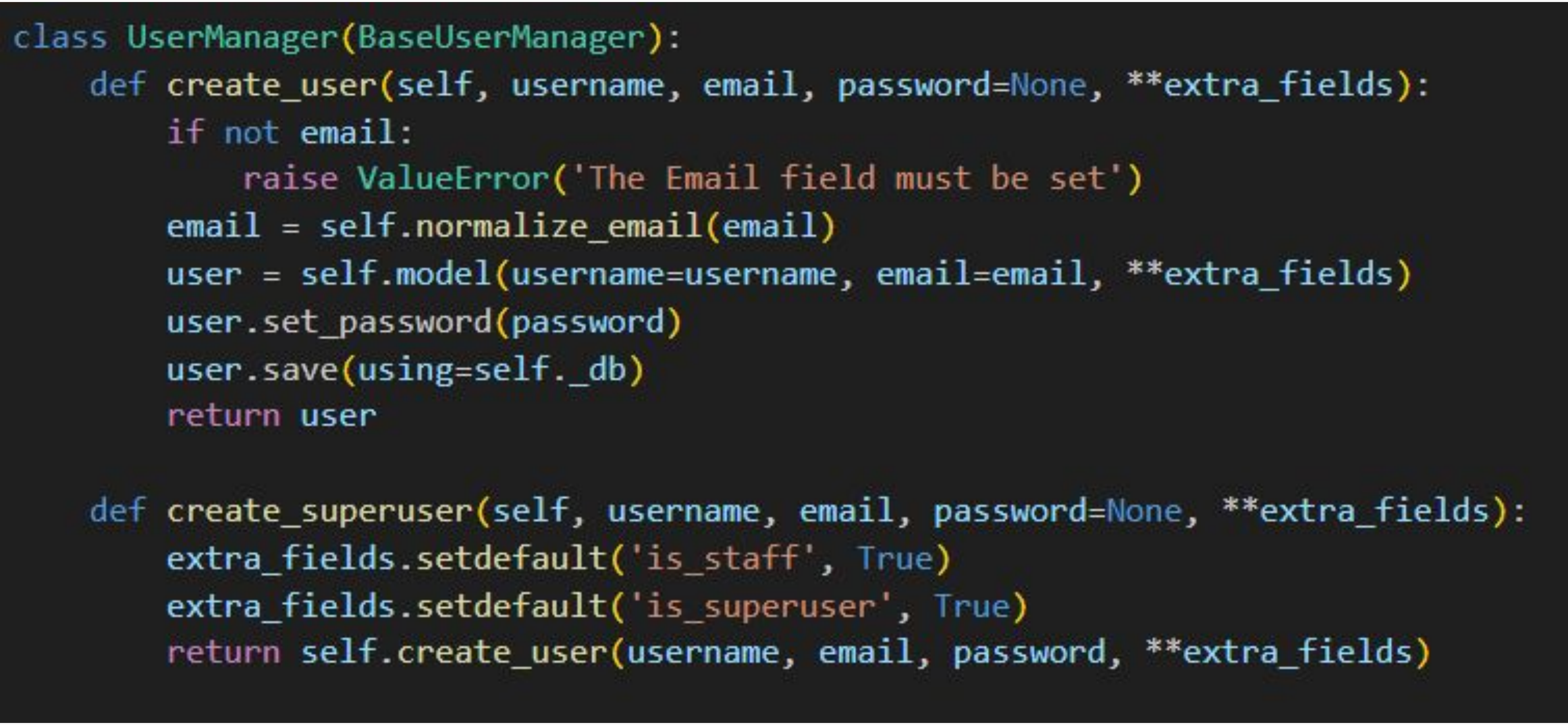
Przykładowe unittesty



```python
    def test_create_parent_consent(self):
        url = reverse('parent_consent_submit', args=[self.consent_template.id])
        data = {
            'child_user': self.student.user_id,
            'is_consent': True
        }
        response = self.client.post(url, data, format='multipart')
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        self.assertEqual(ParentConsent.objects.count(), 1)
        self.assertEqual(ParentConsent.objects.get().is_consent, True)

    def test_get_consent_templates(self):
        self.client.login(username='teacher_test', password='testpass')
        url = reverse('consent_template_list')
        response = self.client.get(url)
        self.assertEqual(response.status_code, status.HTTP_200_OK)

    def test_get_consent_template_detail_teacher(self):
        self.client.login(username='teacher_test', password='testpass')
        url = reverse('consent_template_detail',
                      args=[self.consent_template.id])
        response = self.client.get(url)
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertIn('parent_consents', response.data)

    def test_get_consent_template_detail_parent(self):
        url = reverse('consent_template_detail',
                      args=[self.consent_template.id])
        response = self.client.get(url)
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertNotIn('parent_consents', response.data)
```

```python
    def test_parent_cannot_create_consent_template(self):
        url = reverse('consent_template_list')
        data = {
            'title': 'New Consent',
            'description': 'New Description',
            'end_date': (timezone.now().date() + timedelta(days=10)).isoformat(),
            'students': [self.student.user_id]
        }
        response = self.client.post(url, data, format='json')
        self.assertEqual(response.status_code, status.HTTP_403_FORBIDDEN)

    def test_student_cannot_create_consent_template(self):
        self.client.login(username='student_test', password='testpass')
        url = reverse('consent_template_list')
        data = {
            'title': 'New Consent',
            'description': 'New Description',
            'end_date': (timezone.now().date() + timedelta(days=10)).isoformat(),
            'students': [self.student.user_id]
        }
        response = self.client.post(url, data, format='json')
        self.assertEqual(response.status_code, status.HTTP_403_FORBIDDEN)

    def test_teacher_can_create_consent_template(self):
        self.client.login(username='teacher_test', password='testpass')
        url = reverse('consent_template_list')
        data = {
            'title': 'New Consent',
            'description': 'New Description',
            'end_date': (timezone.now().date() + timedelta(days=10)).isoformat(),
            'students': [self.student.user_id]
        }
        response = self.client.post(url, data, format='json')
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        self.assertEqual(ConsentTemplate.objects.count(), 2)
```

Wzorce projektowe: Factory



```python
class UserManager(BaseUserManager):
    def create_user(self, username, email, password=None, **extra_fields):
        if not email:
            raise ValueError('The Email field must be set')
        email = self.normalize_email(email)
        user = self.model(username=username, email=email, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, username, email, password=None, **extra_fields):
        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)
        return self.create_user(username, email, password, **extra_fields)
```

Dziękujemy za uwagę

Kryspin Kucha
Jakub Konecki
Illia Kuziv