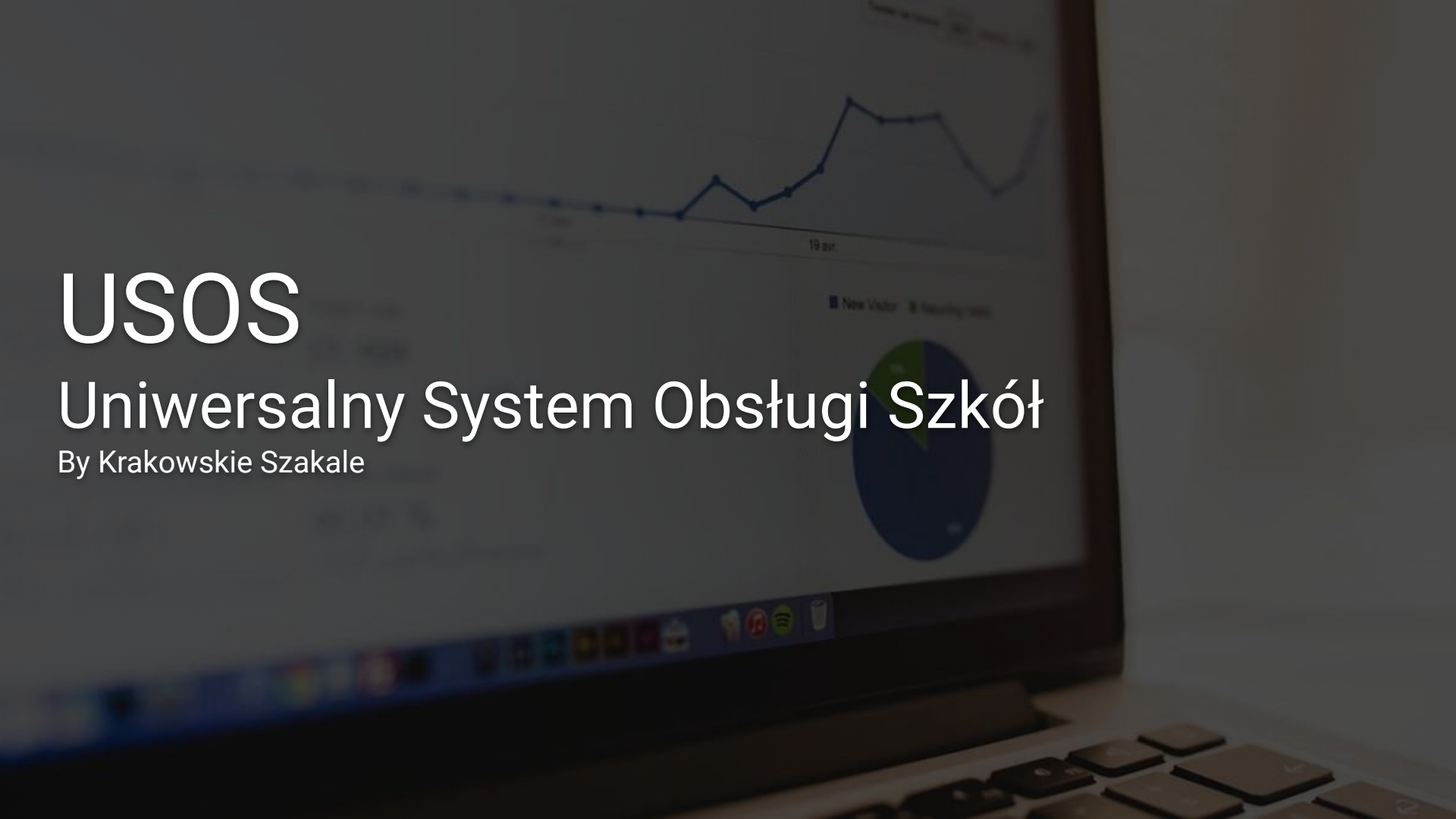


USOS

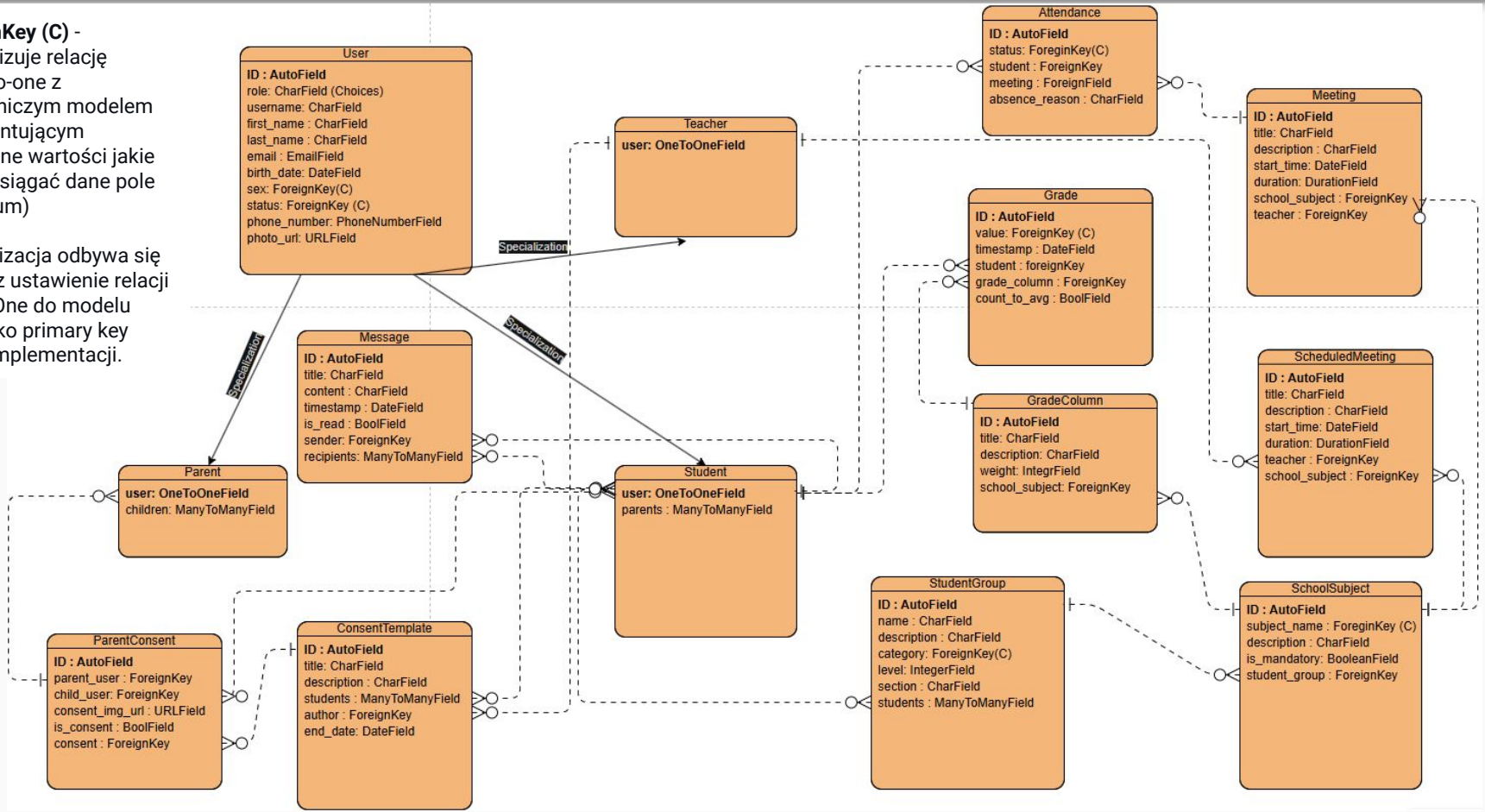
Uniwersalny System Obsługi Szkół

By Krakowskie Szakale

Data model diagram (ERD)

ForeignKey (C) - symbolizuje relację many-to-one z pomocniczym modelem reprezentującym konkretne wartości jakie może osiągać dane pole (jak enum)

Specjalizacja odbywa się poprzez ustawienie relacji OneToOne do modelu User jako primary key danej implementacji.



ForeignKey (C) - przykłady

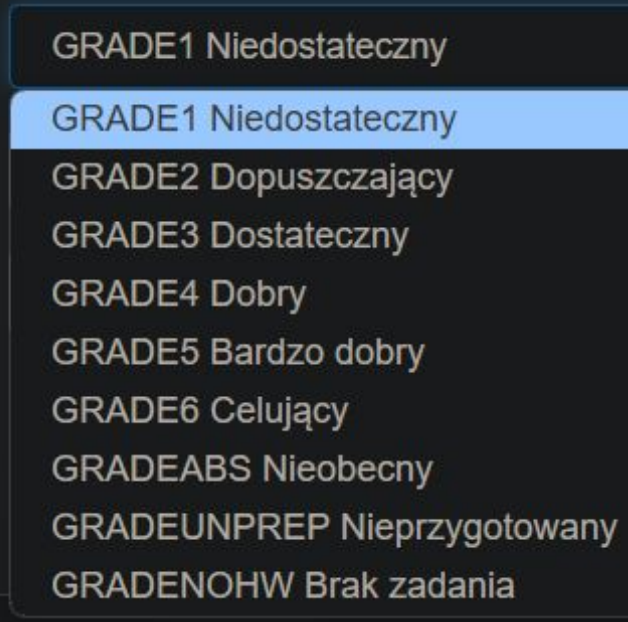
```python
class CategoryStudentGroup(models.Model):
    code = models.CharField(max_length=30, unique=True)
    name = models.CharField(max_length=63,blank=True)

    def __str__(self):
        return f"{self.code} {self.name}"


class CategoryGradeValue(models.Model):
    code = models.CharField(max_length=30, unique=True)
    name = models.CharField(max_length=63,blank=True)

    def __str__(self):
        return f"{self.code} {self.name}"


class CategoryAttendanceStatus(models.Model):
    code = models.CharField(max_length=30, unique=True)
    name = models.CharField(max_length=63,blank=True)

    def __str__(self):
        return f"{self.code} {self.name}"
```

Value

Student

Grade column

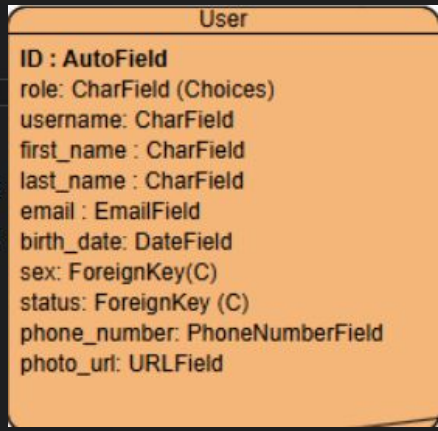
Count to avg



Model: User

```python
class User(AbstractUser):
    """ Username is user's unique identifier (inherited from AbstractUser) """
    ROLE_CHOICES = [
        ('student', 'Student'),
        ('parent', 'Parent'),
        ('teacher', 'Teacher'),
    ]
    role = models.CharField(max_length=10, choices=ROLE_CHOICES, editable=False
    first_name = models.CharField(max_length=255, default="name_example")
    last_name = models.CharField(max_length=255, default="Lname_example")
    email = models.EmailField(unique=True, default="example@example.com")
    birth_date = models.DateField(default=datetime.date(2010, 1, 1))
    sex = models.CharField(max_length=15, choices=[(
        "M", "Male"), ("F", "Female")], default="M")
    status = models.CharField(max_length=31, choices=[(
        "A", "Active"), ("U", "Inactive")], default="A")
    phone_number = models.CharField(max_length=15, blank=True, null=True)
    photo_url = models.URLField(blank=True, null=True)

    objects = UserManager()

    def __str__(self):
        return f"{self.username}: {self.first_name} {self.last_name}"
```

```
GET /user/

HTTP 200 OK
Allow: GET, PUT, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 8,
    "username": "student4",
    "first_name": "student4",
    "last_name": "student4",
    "email": "student4@student4.pl",
    "status": "A",
    "birth_date": "2000-11-11",
    "sex": "M",
    "phone_number": "123123123",
    "photo_url": "",
    "role": "student"
}
```



Modele: Student, Teacher, Parent

```python
class Student(models.Model):
    user = models.OneToOneField(
        User, on_delete=models.CASCADE, primary_key=True)
    parents = models.ManyToManyField(
        "Parent", related_name="children", blank=True)

    def __str__(self):
        return f"Student: {self.user.username}"


class Teacher(models.Model):
    user = models.OneToOneField(
        User, on_delete=models.CASCADE, primary_key=True)

    def __str__(self):
        return f"Teacher: {self.user.username}"


class Parent(models.Model):
    user = models.OneToOneField(
        User, on_delete=models.CASCADE, primary_key=True)

    def __str__(self):
        return f"Parent: {self.user.username}"

```

Modele: StudentGroup, SchoolSubject

```python
class StudentGroup(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True, default="")
    category = models.ForeignKey(
        CategoryStudentGroup, on_delete=models.SET_NULL, null=True)
    level = models.IntegerField()
    section = models.CharField(max_length=50, blank=True, null=True)
    students = models.ManyToManyField(Student, related_name="student_groups")

    def __str__(self):
        return self.name


class SchoolSubject(models.Model):
    subject_name = models.CharField(max_length=255)
    description = models.TextField(blank=True, default="")
    is_mandatory = models.BooleanField(default=False)
    student_group = models.ForeignKey(StudentGroup, on_delete=models.CASCADE)

    def __str__(self):
        return f"{self.subject_name} for {self.student_group}"
```

```
GET /user/8/student/groups/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "name": "4A",
        "description": "Klasa 4a",
        "category": 1,
        "level": 4,
        "section": "A",
        "students": [
            8
        ]
    }
]
```

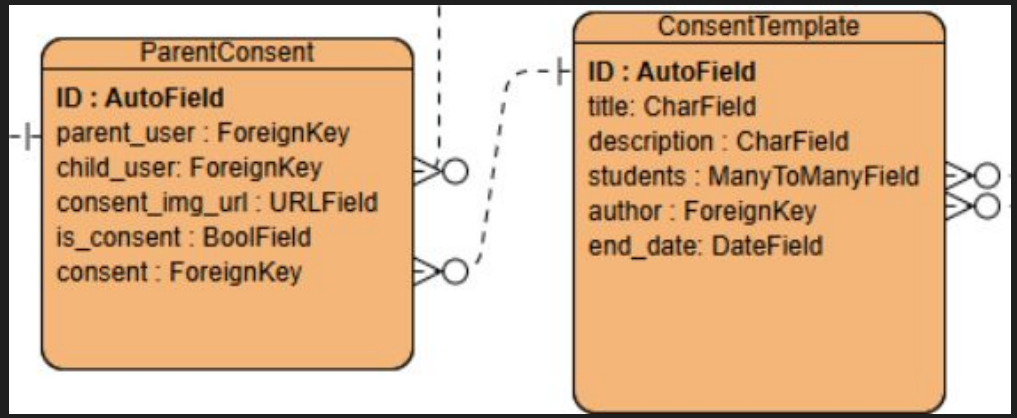
Modele: Grade, GradeColumn

```python
class Grade(models.Model):
    value = models.ForeignKey(
        CategoryGradeValue, on_delete=models.SET_NULL, null=True)
    timestamp = models.DateField(auto_now_add=True)
    student = models.ForeignKey(Student, on_delete=models.CASCADE)
    grade_column = models.ForeignKey("GradeColumn", on_delete=models.CASCADE)
    count_to_avg = models.BooleanField(default=True)

    def __str__(self):
        return f"{self.value} for {self.student}"


class GradeColumn(models.Model):
    title = models.CharField(max_length=255)
    weight = models.IntegerField(default=1)
    description = models.TextField(blank=True, default="")
    school_subject = models.ForeignKey(SchoolSubject, on_delete=models.CASCADE)

    def __str__(self):
        return self.title
```


```
"results": [
    {
        "id": 5,
        "value": "GRADE5",
        "timestamp": "2024-12-04",
        "student": 8,
        "grade_column": 1,
        "count_to_avg": true
    },
    {
        "id": 6,
        "value": "GRADE4",
        "timestamp": "2024-12-04",
        "student": 8,
        "grade_column": 1,
        "count_to_avg": true
    },
    {
        "id": 7,
        "value": "GRADEABS",
        "timestamp": "2024-12-04",
        "student": 8,
        "grade_column": 1,
        "count_to_avg": true
    }
]
```

Modele: ConsentTemplate, ParentConsent

```python
class ConsentTemplate(models.Model):
    author = models.ForeignKey('Teacher', on_delete=models.CASCADE)
    title = models.CharField(max_length=255)
    description = models.TextField(blank=True, default="")
    end_date = models.DateField()
    students = models.ManyToManyField(
        Student, related_name="consent_templates")

    def time_to_end(self):
        return (self.end_date - timezone.now().date()).days

    def is_active(self):
        return timezone.now().date() <= self.end_date

    def __str__(self):
        return f"ConsentTemplate {self.title} by {self.author}
                (Active: {self.is_active()})"


class ParentConsent(models.Model):
    parent_user = models.ForeignKey('Parent', on_delete=models.CASCADE)
    child_user = models.ForeignKey('Student', on_delete=models.CASCADE)
    consent = models.ForeignKey(ConsentTemplate, on_delete=models.CASCADE)
    is_consent = models.BooleanField(default=False)
    url = models.URLField(blank=True, null=True)

    def __str__(self):
        return f"Consent by {self.parent_user} for {self.child_user}"
```



Modele: ScheduledMeeting, Meeting

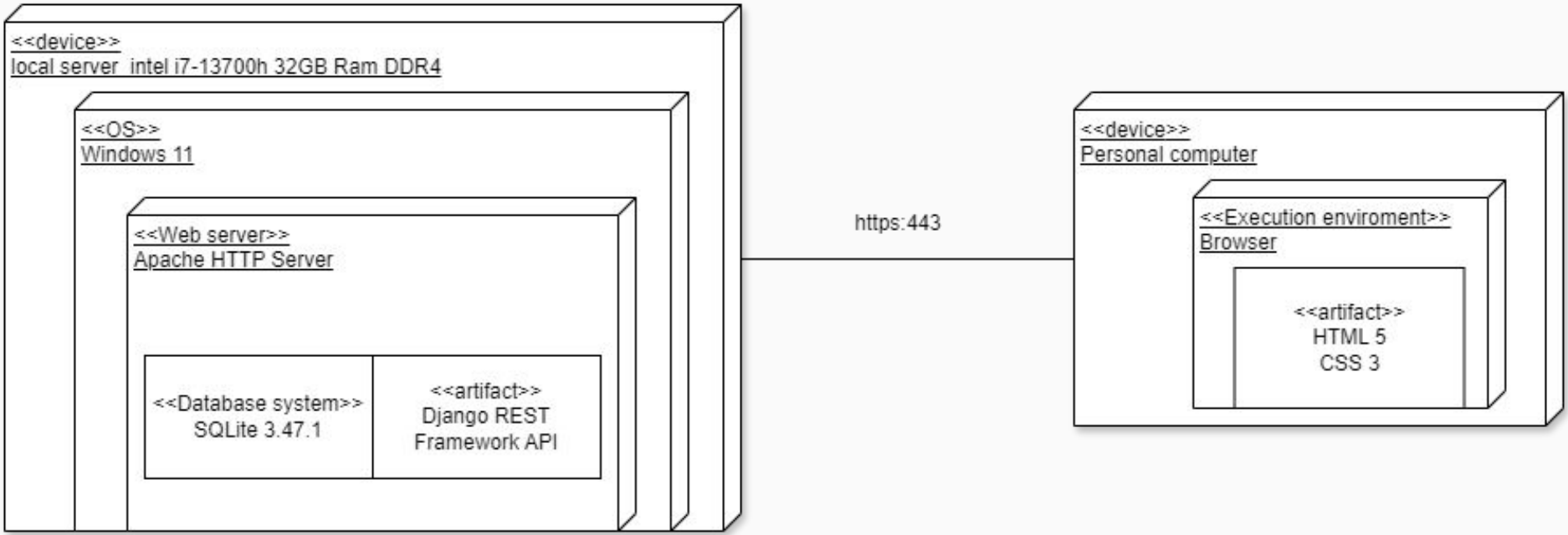
```
198  class ScheduledMeeting(models.Model):
199      title = models.CharField(max_length=255)
200      description = models.TextField(blank=True, default="")
201      start_time = models.DateTimeField()
202      duration = models.DurationField(default=timedelta(
203          minutes=45), validators=[validate_duration])
204      teacher = models.ForeignKey(
205          'Teacher', on_delete=models.CASCADE, related_name='scheduled_meetings')
206      school_subject = models.ForeignKey(
207          'SchoolSubject', on_delete=models.CASCADE)
208
209      def __str__(self):
210          return self.title
211
212
213  class Meeting(models.Model):
214      title = models.CharField(max_length=255)
215      description = models.TextField(blank=True, default="")
216      start_time = models.DateTimeField()
217      duration = models.DurationField(default=timedelta(
218          minutes=45), validators=[validate_duration])
219      teacher = models.ForeignKey('Teacher', on_delete=models.CASCADE)
220      school_subject = models.ForeignKey(
221          'SchoolSubject', on_delete=models.CASCADE)
222
223      def __str__(self):
224          return self.title
```

Przykładowe serializatory

```python
class StudentGroupSerializer(serializers.ModelSerializer):
    class Meta:
        model = StudentGroup
        fields = ['id', 'name', 'description',
                  'category', 'level', 'section', 'student


class SchoolSubjectSerializer(serializers.ModelSerializer)
    class Meta:
        model = SchoolSubject
        fields = ['id', 'subject_name', 'description',
                  'is_mandatory', 'student_group']
class StudentSerializer(serializers.ModelSerializer):
    user = UserSerializer()

    class Meta:
        model = Student
        fields = ['user_id', 'user', 'parents']

    def create(self, validated_data):
        user_data = validated_data.pop('user')
        parents_data = validated_data.pop('parents', [])
        user = User.objects.create_user(**user_data)
        student = Student.objects.create(user=user, **validated_data)
        student.parents.set(parents_data)
        return student
```

```python
class GradeSerializer(serializers.ModelSerializer):
    value = serializers.SlugRelatedField(
        queryset=CategoryGradeValue.objects.all(),
        slug_field="code"
    )
    count_to_avg = serializers.BooleanField(default=True)
    grade_column = serializers.PrimaryKeyRelatedField(
        queryset=GradeColumn.objects.all())

    class Meta:
        model = Grade
        fields = ['id', 'value', 'timestamp',
                  'student', 'grade_column', 'count_to_avg']

    def create(self, validated_data):
        grade = Grade.objects.create(
            grade_column=validated_data.pop('grade_column'), **validated_data)
        return grade
```

Deployment diagram




Dziękujemy za uwagę
Kryspin Kucha
Jakub Konecki
Illia Kuziv