МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра Систем информатики

Направление подготовки 09.06.01 – Информатика и вычислительная техника

ОТЧЕТ

Обучающегося Козубенко Андрея Алексеевича **группы № 22215 курса 4**
(Ф.И.О. полностью)

Тема задания: Разработка хранилища корпуса текстов

Новосибирск 2025

Оглавление

Введение

Целью данной лабораторной работы является разработка REST-хранилища корпуса текстов на языке **Python** с использованием фреймворка **Django**.

Задачи лабораторной работы:

- разработать модели данных для корпуса и текстов;
- реализовать репозитории для работы с таблицами корпуса и текстов (CRUD-операции);
- создать REST-эндпоинты для редактирования данных;
- реализовать интеграцию с репозиторием онтологий (OntologyRepository);
- протестировать работу эндпоинтов с помощью Postman.

Реализация хранилища корпуса

Хранилище реализовано в виде Django-приложения db.
Созданы две основные модели:

1. **Corpus**
   - title — название корпуса;
   - description — краткое описание;
   - genre — жанр текстов.
2. **Text**
   - title — название текста;
   - description — описание текста;
   - text — содержимое текста (TextField);
   - corpus — связь с моделью Corpus;
   - has_translation — связь с самим собой для хранения перевода.

Для взаимодействия с базой данных созданы классы-репозитории:
CorpusRepository
Реализует CRUD-операции для корпуса:

- create_corpus(title, description, genre) — создание корпуса;
- update_corpus(id, **kwargs) — обновление данных;
- get_corpus(id) — получение корпуса вместе с текстами;
- delete_corpus(id) — удаление корпуса.

TextRepository
Реализует CRUD-операции для текстов:

- create_text(title, description, text, corpus_id, has_translation);
- update_text(id, **kwargs);
- get_text(id);
- delete_text(id).

Для доступа к репозиториям реализованы REST-эндпоинты (в views.py):

Операция	**Метод**	**URL**	**Описание**
Create corpus	POST	/api/corpus/create/	Создание корпуса
Update corpus	POST	/api/corpus/update/	Изменение данных корпуса
Get corpus	GET	/api/corpus/?id=1	Получение корпуса и его текстов
Delete corpus	DELETE	/api/corpus/delete/?id=1	Удаление корпуса
Create text	POST	/api/text/create/	Добавление текста
Update text	POST	/api/text/update/	Редактирование текста
Get text	GET	/api/text/?id=1	Получение текста
Delete text	DELETE	/api/text/delete/?id=1	Удаление текста

Реализация репозитория онтологий

Для интеграции с графовой базой данных Neo4j добавлен модуль OntologyRepository.
Он наследуется от Neo4jRepository и реализует управление структурами онтологий, включая классы, объекты и свойства.

Основные группы методов:

1. **Работа с онтологией:**
   - get_ontology() — получить все классы и связи;
   - get_class(uri) — получить конкретный класс;
   - get_class_children(), get_class_parents() — получить потомков и родителей.
2. **Управление классами:**
   - create_class(), update_class(), delete_class().
3. **Атрибуты классов:**
   - add_class_attribute() и delete_class_attribute() — DatatypeProperty;
   - add_class_object_attribute() и delete_class_object_attribute() — ObjectProperty.
4. **Объекты классов:**
   - create_object(), update_object(), get_object(), delete_object().
5. **Сигнатуры классов:**
   - collect_signature() — получение всех свойств класса и связей.

Также реализованы эндпоинты:

GET /ontology/
GET /ontology/class/
POST /ontology/class/create/
POST /ontology/class/update/
DELETE /ontology/class/delete/
POST /ontology/object/create/
POST /ontology/object/update/
DELETE /ontology/object/delete/
GET /ontology/signature/

Реализация

https://github.com/andrew-kozubenko/Linguistics/text_corpus_rep

Листинги

Листинг 1 – Модель корпуса и текста (models.py):

```
class Corpus(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()
    genre = models.CharField(max_length=100)
```

```
class Text(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()
    text = models.TextField()
    corpus = models.ForeignKey(Corpus, on_delete=models.CASCADE, related_name="texts")
    has_translation = models.ForeignKey(
        to: 'self',
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
        related_name='translations'
    )
```

Листинг 2 – Пример метода создания текста:

```
def create_text(self, title, description, text, corpus_id, has_translation=None):
    corpus = Corpus.objects.get(id=corpus_id)
    t = Text.objects.create(
        title=title,
        description=description,
        text=text,
        corpus=corpus,
        has_translation=has_translation
    )
    return self.collect_text(t)
```

Листинг 3 – Пример API-эндпоинта:

```
@api_view(['GET'])
@permission_classes((AllowAny,))
def getCorpus(request):
    corpus_id = request.GET.get("id")
    repo = CorpusRepository()
    result = repo.get_corpus(corpus_id)
    return Response(result)
```

Заключение

В ходе выполнения лабораторной работы были реализованы:

1. Модели Corpus и Text для хранения текстов предметной области;
2. Репозитории CorpusRepository и TextRepository с полным набором CRUD-операций;
3. REST-эндпоинты для взаимодействия с данными через Django;
4. Репозиторий OntologyRepository для управления онтологиями и интеграции с Neo4j;
5. Тестирование всех запросов через Postman.

Все задачи, поставленные во введении, были успешно выполнены.

Список литературы

1. Django documentation: https://docs.djangoproject.com/
2. Neo4j documentation: https://neo4j.com/docs
3. REST framework: https://www.django-rest-framework.org/