**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 3](#_8xl0k4o7g53y)

[Реализация хранилища корпуса 4](#_18lqawz87498)

[Листинги](#_18lqawz87498) 6

[Заключение](#_ftfmarjidp06) 7

[Список литературы](#_8i44esq2u94t) 8

# 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/

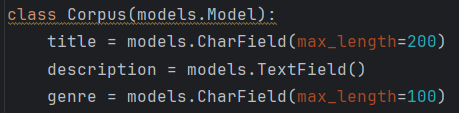
GET /ontology/signature/

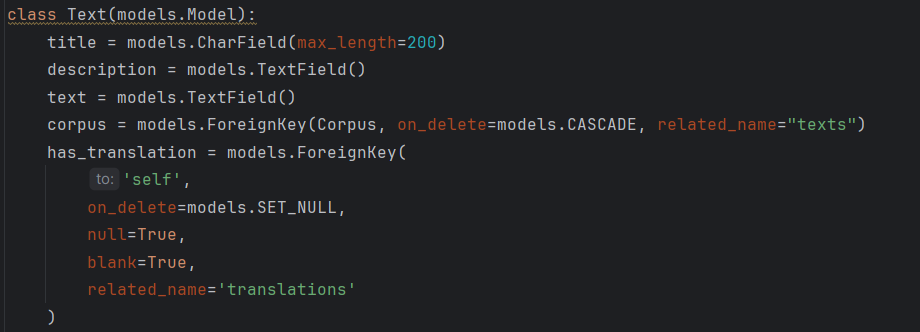
**Реализация**

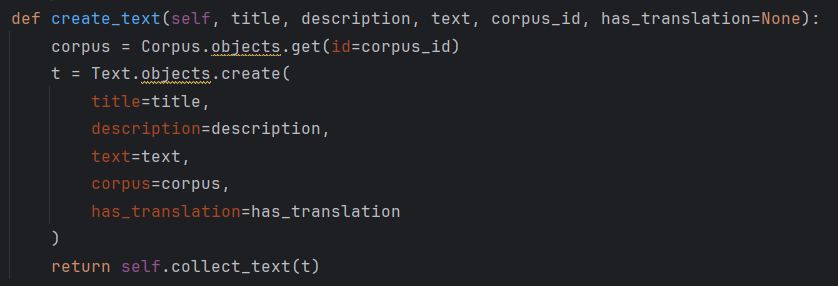
[**https://github.com/andrew-kozubenko/Linguistics/text\_corpus\_rep**](https://github.com/andrew-kozubenko/Linguistics/text_corpus_rep)

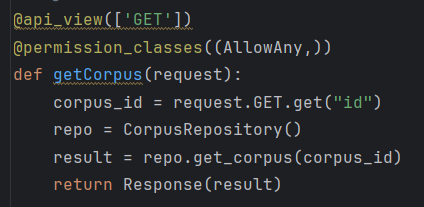
# Листинги

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



**Листинг 2 – Пример метода создания текста:**

**Листинг 3 – Пример API-эндпоинта:**



# 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/