Цели практической работы

- Научиться настраивать приложение для работы с базой данных PostgreSQL.
- Научиться управлению версиями базы данных с помощью Liquibase.
- Освоить использование Spring JPA для настройки связей между сущностями в веб-приложении.

Что входит в практическую работу

1. Расширение функциональности приложения, которое мы создавали.
2. Добавление сущности «Категории» для классификации новостей. Каждая новость будет принадлежать к одной категории.
3. Разработка API для управления категориями новостей.
4. Переход от хранения данных в памяти (с использованием ConcurrentHashMap) к хранению в постоянной базе данных PostgreSQL.
5. Создание таблиц News и Category с помощью Liquibase для хранения данных о новостях и категориях.

Что нужно сделать

Выполните задание:

1. Запустите базу данных PostgreSQL.
2. Добавьте в проект зависимость <spring-boot-starter-data-jpa> для работы с базами данных.
3. Добавьте зависимость драйвера <postgresql>.
4. Добавьте зависимость <liquibase-core>.
5. Создайте сущность Category для отображения категории новостей. Эта сущность должна содержать следующие поля:
    - id (тип Long) — уникальный идентификатор новости;
    - title (тип String) — название категории.
6. Обновите класс News и добавьте связь с Category. Каждая новость должна принадлежать к одной категории. Для этого вы можете использовать аннотацию @ManyToOne.
7. Перейдите от хранения данных в памяти (с использованием ConcurrentHashMap) к хранению в постоянной базе данных PostgreSQL. Для этого добавьте настройки подключения к базе данных в файле application.properties.
8. Используя Liquibase, создайте через файлы миграции две таблицы в базе данных для сущностей News и Category. В таблице News должно быть поле для связи с категорией (например, category_id). Используйте SQL синтаксис для миграций.
9. Создайте API для управления категориями, включая следующие методы:

- Получение списка всех категорий:

URL: /api/categories/{id}

HTTP-метод: GET

Описание: Получение информации о категории по её уникальному идентификатору.

Параметры пути:

id (Long): Уникальный идентификатор категории.

Ответ:

Код HTTP-ответа 200 (OK) и объект Category в формате JSON с информацией о новости.

```
{
 "id": 1,
 "title": "Web-разработка"
}
```

Код HTTP-ответа 404 (Not Found), если категории с указанным id не найдена.

```
{
 "message": "Категория с id 1 не найдена."
}
```

- Получение всех категорий в формате JSON:

URL: /api/categories

HTTP-метод: GET

Описание: Получение списка всех категорий по её уникальному идентификатору.

Ответ:

Код HTTP-ответа 200 (OK) и массив объектов Category с информацией о категориях.

```
[
 {
 "id": 1,
 "title": "Web-разработка"

 },
 {
 "id": 2,
 "title": "Нейросети"
 }
]
```

- Создание новой категории:

URL: /api/categories

HTTP-метод: POST

Описание: Создание новой категории.

Запрос: Объект Category в формате JSON с заполненными полями title.

```
{
 "title": "Дизайн",
}
```

Ответ:

Код состояния HTTP 201 (Created) и объект Category в формате JSON с информацией о созданной категории.

```
{
 "id": 3,
 "title": "Дизайн"
}
```

- Обновление существующей категории:

URL: /api/categories

HTTP-метод: PUT

Описание: Обновление существующей категории.

Запрос: Объект Category в формате JSON с заполненными полями id, title.

```
{
 "id": 3,
 "title": "Web-дизайн"
}
```

Ответ:

Код HTTP-ответа 200 (OK) и объект Category в формате JSON с информацией об обновлённой категории.

```
{
 "id": 3,
 "title": "Web-дизайн"
}
```

- Удаление категории по ID:

URL: /api/categories/{id}

HTTP-метод: DELETE

Описание: Удаление категории по её уникальному идентификатору.

Параметры пути: id (Long): Уникальный идентификатор категории.

Ответ:

Код HTTP-ответа 200 (OK) в случае успешного удаления категории.

Код HTTP-ответа 404 (Not Found), если категория с указанным id не найдена, и отправка сообщения.

```
{
 "message": "Категория с id 3 не найдена."
}
```

10. Обновите API для управления новостями с учётом связи с категориями:

- Получение новости по ID:

URL: /api/news/{id}

HTTP-метод: GET

Описание: Получение информации о новости по её уникальному идентификатору. Добавить в ответ категорию.

Параметры пути:

id (Long): Уникальный идентификатор новости.

Ответ:

Код HTTP-ответа 200 (OK) и объект News в формате JSON с информацией о новости.

```
{
 "id": 1,
 "title": "Заголовок новости",
 "text": "Текст новости",
 "date": "2023-10-20T12:00:00Z"

 "category": "Web-разработка”
}
```

Код HTTP-ответа 404 (Not Found), если новость с указанным id не найдена.

```
{
 "message": "Новость с id 1 не найдена."
}
```

- Получение списка новостей:

URL: /api/news

HTTP-метод: GET

Описание: Получение списка всех новостей в формате JSON. Добавить в ответ категорию.

Ответ:

Код HTTP-ответа 200 (OK) и массив объектов News с информацией о новостях.

```
[
 {
 "id": 1,
 "title": "Заголовок новости 1",
 "text": "Текст новости 1",
 "date": "2023-10-20T12:00:00Z",

 "category": "Web-разработка”
 },
 {
 "id": 2,
 "title": "Заголовок новости 2",
 "text": "Текст новости 2",
 "date": "2023-10-21T10:30:00Z",

 "category": "Дизайн”
 }
]
```

- Создание новой новости с категорией:

URL: /api/news

HTTP-метод: POST

Описание: Создание новой новости. Добавить поле категории.

Запрос: Объект News в формате JSON с заполненными полями title, text и date.

```
{
 "title": "Новая новость",
 "text": "Текст новости",

 "category": "Нейросети”
}
```

Ответ:

Код HTTP-ответа 201 (Created) и объект News в формате JSON с информацией о созданной новости.

```
{
 "id": 3,
 "title": "Новая новость",
 "text": "Текст новости",
 "date": "2023-10-22T14:45:00Z"

 "category": "Нейросети”
}
```

- Обновление существующей новости с категорией:

URL: /api/news

HTTP-метод: PUT

Описание: Обновление существующей новости. Добавить поле категории.

Запрос: Объект News в формате JSON с заполненными полями id, title, text и date.

```
{
 "id": 3,
 "title": "Обновлённая новость",
 "text": "Обновлённые новости",

 "category": "Нейросети”
}
```

Ответ:

Код HTTP-ответа 200 (OK) и объект News в формате JSON с информацией об обновлённой новости.

```
{
 "id": 3,
 "title": "Обновлённая новость",
 "text": "Обновлённые новости",
 "date": "2023-10-22T14:55:00Z",

 "category": "Нейросети”
}
```

- Получение новостей по идентификатору категории:

URL: /api/news/category/{id}

HTTP-метод: GET

Описание: Получение списка всех новостей в формате JSON, относящихся к указанной категории.

Параметры пути:

id (Long): Уникальный идентификатор категории.

Ответ:

Код HTTP-ответа 200 (OK) и массив объектов News с информацией о новостях.

```
[
 {
 "id": 1,
 "title": "Заголовок новости 1",
 "text": "Текст новости 1",
 "date": "2023-10-20T12:00:00Z",

 "category": "Web-разработка”
 },
 {
 "id": 4,
 "title": "Заголовок новости 2",
 "text": "Текст новости 2",
 "date": "2023-10-21T10:30:00Z",

 "category": "Web-разработка”
 }
]
```

Что оценивается

- Проект Spring Boot создан и настроен согласно вводным данным задания.
- Создан контроллер API с методами для выполнения операций над категориями (GET, POST, PUT, DELETE).
- Созданы сервисы и репозитории для работы с сущностями News и Category.
- Настроена связь между сущностями News и Category.
- Миграция структуры базы данных производится с помощью Liquibase.
- Приложение подключено к базе данных PostgreSQL.
- Весь функционал работы с категориями новостей и новостями работает через API.

Критерии оценки

Принято: методы API корректно выполняют операции и возвращают соответствующие ответы и статусы HTTP, задание выполнено.