

Техническое задание

Реализовать веб-сайт для статей на разные темы. Автор статьи может редактировать и удалять статью. Админ может удалять статьи. Все авторизованные пользователи могут просматривать статью, добавлять комментарии, ставить оценку статье.

У авторизованного пользователя есть доступ к его персональной странице, на которой он может просматривать список статей, которые он опубликовал, перейти на страничку добавления статьи. Так же на странице профиля отображаются его основные данные: никнейм, имейл, дата регистрации. Также на странице пользователя есть дополнительные данные о пользователе. Это может быть номер телефона, пол и т.д. Типы таких данных может указывать (создавать), а также удалять администратор. Пользователь имеет возможность редактирования дополнительных данных о себе.

Авторизованные пользователи имеют возможность добавлять, удалять и редактировать комментарии. Администраторы имеют возможность удалять комментарии любых пользователей.

В базе данных хранится количество посещений для каждой статьи, оно увеличивается при посещении статьи как аутентифицированным, так и не аутентифицированным пользователем.

Для каждой статьи необходимо загрузить картинку, которая, в последствии, будет расположена в заголовке к статье. Картинки могут храниться как на облаке, так и в базе данных, в зависимости от выбора администратора.

Авторизованные пользователи имеют возможность оценивать статью при помощи кнопок. Оценка может иметь значение "LIKE" и "DISLIKE".

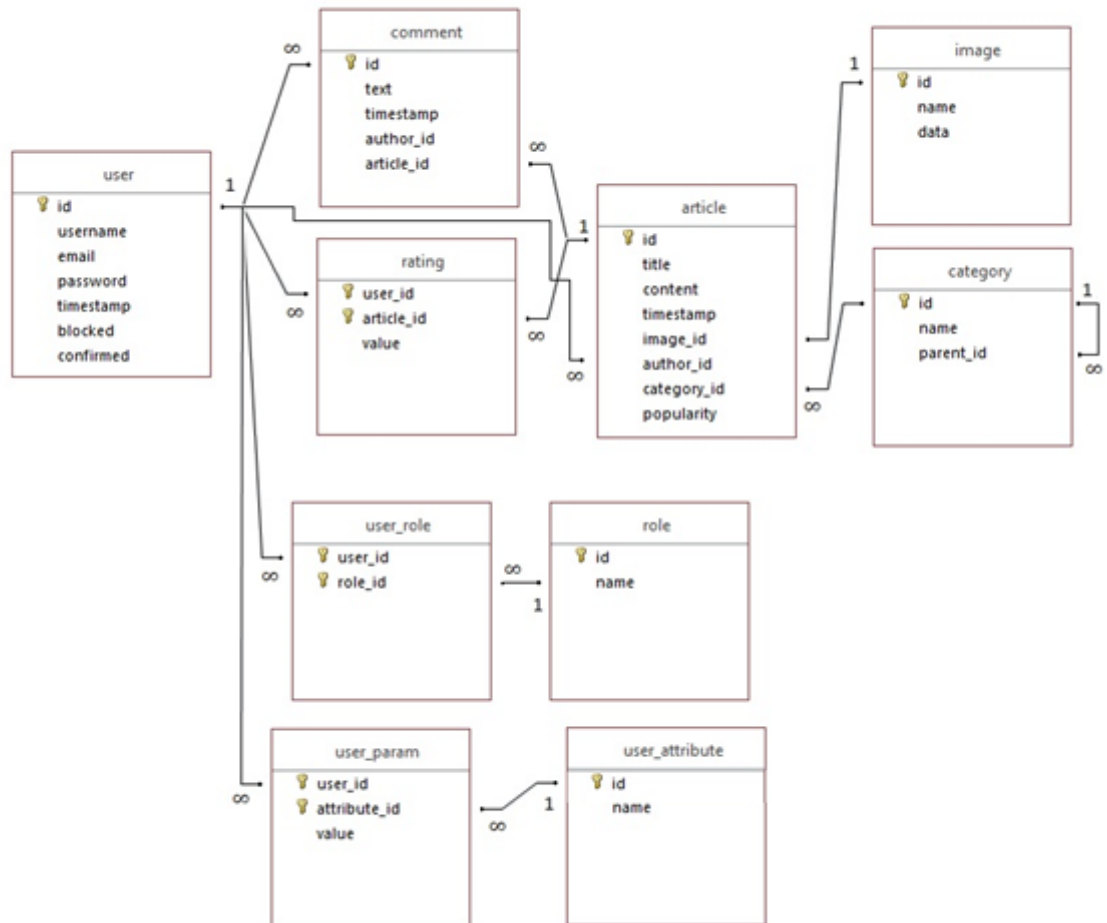
При создании статьи необходимо указать категорию, к которой она относится. Категории могут образовывать древовидную иерархию. Админ может создавать и удалять категории.

Админам доступна страница «Администрирование», где можно удалить, заблокировать, сделать админом и разжаловать любого пользователя. Также админам доступен SQL-терминал.

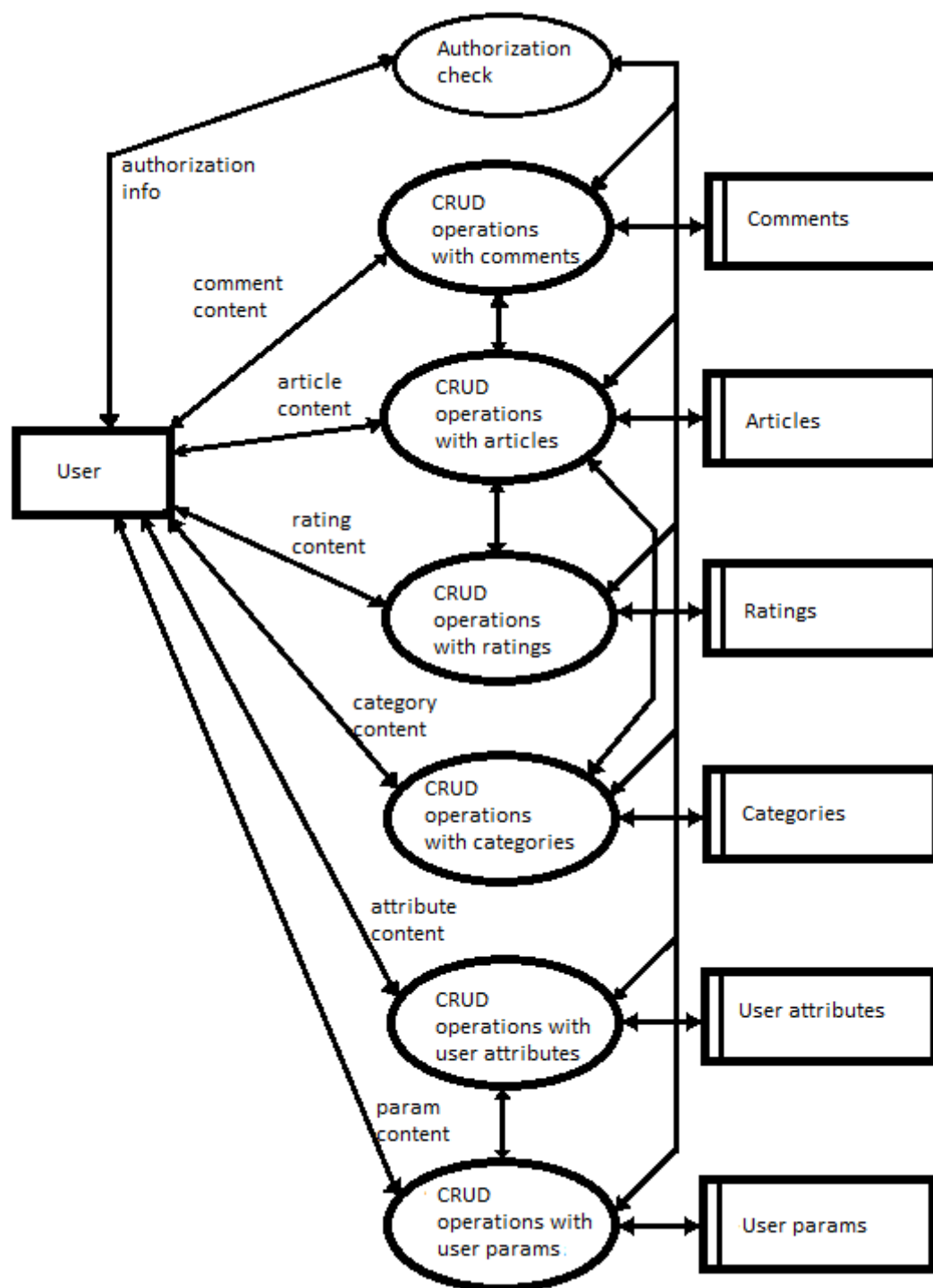
Статью можно скачивать в формате pdf, нажав на соответствующую кнопку.

Ссылка на проект: <https://github.com/yatskevichfyodor/articles>

ER – диаграмма

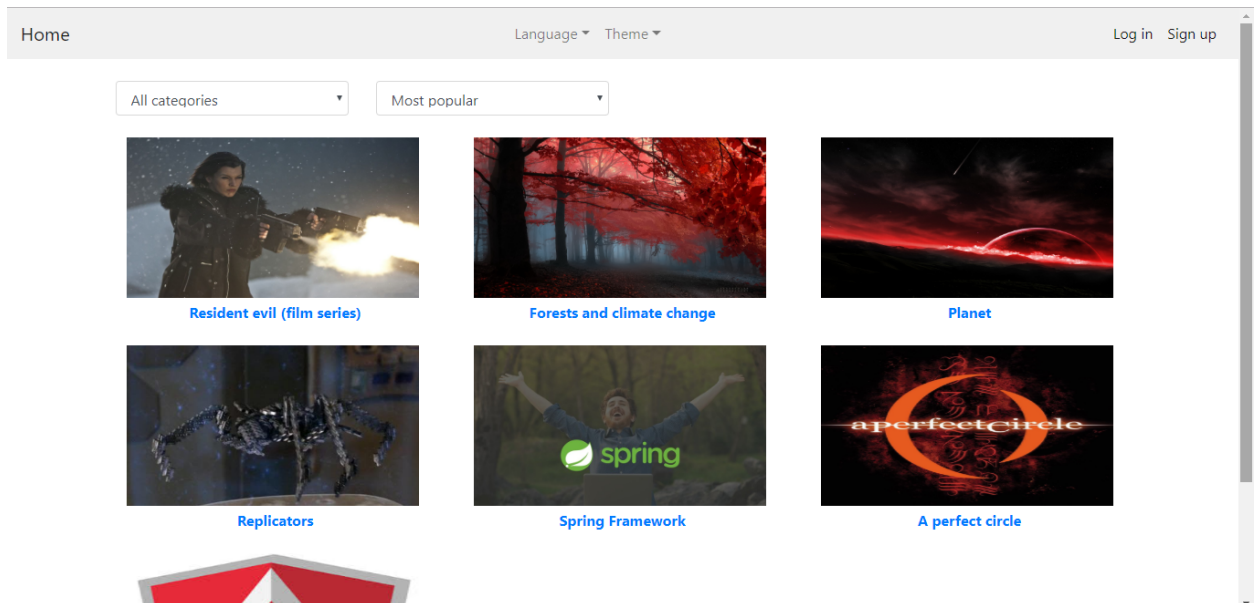


DF – диаграмма



Описание основной бизнес-логики

Главная страница



Здесь имеется возможность отфильтровать статьи, а также выбрать порядок вывода статей (По популярности – по убыванию, по дате добавления – по возрастанию и убыванию). При выборе категории выводятся статьи для выбранной категории, а также для всех подкатегорий.

Код, отвечающий за получение статей в базе данных отличается в зависимости от выбора порядка вывода и может быть представлен одним из следующих методов:

```
public List<Article> getArticlesByCategoriesSortedByPopularity(List<Category> categories) throws SQLException {
    if (categories.size() == 0) return null;

    StringBuilder statement = new StringBuilder(" " +
        "SELECT *\n" +
        "FROM article a\n" +
        "WHERE\n" +
        "    a.category_id in (");
    for (Category category: categories) {
        statement.append(category.getId());
        statement.append(", ");
    }
    statement.delete(statement.length() - 2, statement.length());
    statement.append(")\n" +
        "ORDER BY popularity DESC;");

    PreparedStatement ps = connect.prepareStatement(statement.toString());
    ResultSet rs = ps.executeQuery();

    List<Article> result = new ArrayList<>();
    while (rs.next()) {
        result.add(convertResultSetToArticle(rs));
    }

    if (ps != null)
        ps.close();

    return result;
}
```

```

public List<Article> getArticlesByCategoriesSortedByDateDesc(List<Category> categories) throws SQLException {
    StringBuilder statement = new StringBuilder(" " +
        "SELECT *\n" +
        "FROM article a\n" +
        "WHERE\n" +
        "    a.category_id in (");
    for (Category category: categories) {
        statement.append(category.getId());
        statement.append(", ");
    }
    statement.delete(statement.length() - 2, statement.length());
    statement.append(")\n" +
        "ORDER BY timestamp DESC;");

    PreparedStatement ps = connect.prepareStatement(statement.toString());
    ResultSet rs = ps.executeQuery();

    List<Article> result = new ArrayList<>();
    while (rs.next()) {
        result.add(convertResultSetToArticle(rs));
    }

    if (ps != null)
        ps.close();

    return result;
}

```

```

public List<Article> getArticlesByCategoriesSortedByDateAsc(List<Category> categories) throws SQLException {
    StringBuilder statement = new StringBuilder(" " +
        "SELECT *\n" +
        "FROM article a\n" +
        "WHERE\n" +
        "    a.category_id in (");
    for (Category category: categories) {
        statement.append(category.getId());
        statement.append(", ");
    }
    statement.delete(statement.length() - 2, statement.length());
    statement.append(")\n" +
        "ORDER BY timestamp ASC;");

    PreparedStatement ps = connect.prepareStatement(statement.toString());
    ResultSet rs = ps.executeQuery();

    List<Article> result = new ArrayList<>();
    while (rs.next()) {
        result.add(convertResultSetToArticle(rs));
    }

    if (ps != null)
        ps.close();

    return result;
}

```

Каждый из методов принимает в качестве параметров список категорий, по которым выбираются статьи для вывода. Данные категории получаются при помощи вызова двух методов.

Первый получает иерархию использованных категорий, а второй преобразует иерархию в лист.

Для формирования иерархии использованных категорий используется класс UsedCategoriesHierarchyBuilder. Формирование происходит при помощи следующего кода:

```
private void build() {
    usedCategories = new HashSet<>(categoryRepository.findUsedCategories());
    usedCategoriesAndParentsSet = new HashSet<>(usedCategories);
    for (Category category : usedCategories) {
        insertParentCategories(category);
    }

    Set<Category> rootCategories = new HashSet<>();
    for (Category category : usedCategoriesAndParentsSet) {
        if (category.getParentCategory() == null)
            rootCategories.add(category);
    }

    hierarchy.setSubcategories(rootCategories);

    for (Category rootCategory: hierarchy.getSubcategories()) {
        rootCategory.setParentCategory(hierarchy);
    }

    usedCategoriesAndParentsList = new ArrayList<>();
    usedCategoriesAndParentsList.addAll(rootCategories);

    for (Category category : hierarchy.getSubcategories()) {
        filterSubcategories(category);
    }
}
```

В данном методе, сначала из БД считываются используемые категории, при помощи следующего класса:

```

public interface CategoryRepository extends JpaRepository<Category, Long> {

    Category findByName(String name);

    @Query(value = "\n" +
        "SELECT *\n" +
        "FROM category c\n" +
        "WHERE\n" +
        "\t(\n" +
        "\t\tSELECT COUNT(*)\n" +
        "\t\tFROM article a\n" +
        "\t\tWHERE a.category_id = c.id\n" +
        "\t) > 0", nativeQuery=true)
    List<Category> findUsedCategories();

    @Query(value = "\n" +
        "SELECT *\n" +
        "FROM\n" +
        "category c\n" +
        "WHERE\n" +
        "c.id IN\n" +
        "(SELECT\n" +
        "a.category_id\n" +
        "FROM\n" +
        "article a\n" +
        "WHERE\n" +
        "a.author_id = :id\n" +
        ")", nativeQuery=true)
    List<Category> findUsedCategoriesByUserId(@Param("id") Long id);
}

```

Страница просмотра статьи

Заголовок статьи:


Home
Language
Theme
Log in
Sign up

Forests and climate change

Author: [username2](#)

Creating date and time: 2018-05-13 17:34

Category: **Nature**



Forests are essential to our survival and well-being. Forests clean our air, our water, our soil and they regulate our climate, amongst many other things. Trees and forests are not always associated with urban landscapes. However, there too they provide invaluable, often invisible, services. Simply by acting as 'green oasis' in our concrete jungles, they offer recreation and health services for many European citizens. How many of us love strolling through parks and green spaces in cities, tending our gardens and filling our homes with green plants? Access to green environments makes us happier and our bodies healthier. Scientific studies show that urban forests and green spaces help improve physical health and mental well-being. With more than three quarters of Europeans living in urban areas, trees, forests and green spaces mean more than ever before. Climate change increases health risks Climate change projections foresee an increase of 2 to 5 °C by 2100 in mean annual temperatures in Europe. The greatest warming is expected in eastern and northern Europe in winter and in

Нижняя часть: рейтинг, комментарии, кнопка получения статьи в pdf формате.



Для получения при загрузке страницы оценки пользователя и суммарной оценки статьи используется следующий класс:

```
public interface RatingRepository extends JpaRepository<Rating, Long> {  
    @Query(value = "SELECT * FROM rating r where r.user_id = :user_id and r.article_id = :article_id", nativeQuery=true)  
    Rating findByIdAndArticleId(@Param("user_id") Long userId, @Param("article_id") Long articleId);  
  
    void deleteById(Rating.RatingId ratingId);  
  
    @Query(value = "SELECT COUNT(*) FROM rating where article_id = :article_id and value = :value", nativeQuery=true)  
    Long getValuesNumberByArticleId(@Param("article_id") String id, @Param("value") String value);  
  
    @Modifying  
    @Query("DELETE FROM Rating c WHERE c.id in ?1")  
    @Transactional  
    void deleteRatings(@Param("id") Set<Rating.RatingId> id);  
}
```

Для получения списка комментариев для статьи, а также других операций с комментариями используется следующий класс:


```

public interface CommentRepository extends JpaRepository<Comment, Long> {

    @Query(value = "SELECT * FROM comment c where c.article_id = :article_id", nativeQuery=true)
    List<Comment> findByArticleId(@Param("article_id") Long id);

    @Transactional
    void deleteById(@Param("id") Long id);

    @Modifying
    @Query("DELETE FROM Comment c WHERE c.id in ?1")
    @Transactional
    void deleteComments(@Param("id") Set<Long> id);
}

```

Страница пользователя

Home
Language
Theme
Admin page
username1
Log out

username1

Your articles

Add article

- science fiction





Replicators

Basic data

Username	username1
Email	yatskevichfyodor@gmail.com
Password	*****
Registration date & time	2018-01-01 00:00

Additional data

Add field

First name	Fyodor	 
Last name	Yatskevich	 

Для формирования списка категорий для элемента dropdown используются классы UserCategoriesHierarchyBuilder и HierarchicalCategoryHierarchyToListConverter. Принцип формирования иерархии использованных категорий в UserCategoriesHierarchyBuilder аналогичен методу в UsedCategoriesHierarchyBuilder:

```
public void build(User user) {
    usedCategories = new HashSet<>(categoryService.findUsedCategoriesByUser(user));
    usedCategoriesAndParentsSet = new HashSet<>(usedCategories);
    for (Category category : usedCategories) {
        insertParentCategories(category);
    }

    Set<Category> rootCategories = new HashSet<>();
    for (Category category : usedCategoriesAndParentsSet) {
        if (category.getParentCategory() == null)
            rootCategories.add(category);
    }

    hierarchy.setSubcategories(rootCategories);

    for (Category rootCategory: hierarchy.getSubcategories()) {
        rootCategory.setParentCategory(hierarchy);
    }

    usedCategoriesAndParentsList = new ArrayList<>();
    usedCategoriesAndParentsList.addAll(rootCategories);

    for (Category category : hierarchy.getSubcategories()) {
        filterSubcategories(category);
    }
}
```

Страница добавления статьи

[Home](#) [Language ▾](#) [Theme ▾](#) [Admin page](#) [username1](#) [Log out](#)

Article creating

Drop image here or click to upload.

Select category
science ▾ [Category management](#)

Article title

Article content

Create article

Страница управления категориями

[Home](#) [Language ▾](#) [Theme ▾](#) [Admin page](#) [username1](#) [Log out](#)

Category management

Add category

Without parent category ▾

Add category

Delete category

Select category ▾

Delete category

Здесь для формирования категорий для левого элемента dropdown используются классы `UsedCategoryHierarchyBuilder` и `HierarchicalCategoryHierarchyToListConverter`. При помощи `HierarchicalCategoryHierarchyToListConverter`, иерархия не только преобразуется в лист, но еще и обрезается, чтобы максимальный уровень вложенности выводимых категорий не превышал 4. Т.к. максимальный уровень вложенности категории 5, а данный dropdown служит для отображения списка возможных родительских категорий.

```

public List<Category> convert(Category rootCategory, int maxNestingLevel) {
    this.maxNestingLevel = maxNestingLevel;
    for (Category subcategory: rootCategory.getSubcategories()) {
        makeList(subcategory, nestingLevel: 0);
    }

    return list;
}

private void makeList(Category c, int nestingLevel) {
    if (nestingLevel > maxNestingLevel) return;

    StringBuilder newName = new StringBuilder();
    for (int i = 0; i < nestingLevel; i++) {
        newName.append("-");
    }
    newName.append(" ");
    newName.append(c.getName());

    Category category = new Category();
    category.setId(c.getId());
    category.setName(newName.toString());

    list.add(category);

    for (Category subcategory: c.getSubcategories()) {
        makeList(subcategory, nestingLevel: nestingLevel + 1);
    }
}

```

При удалении категории происходит валидация на наличие подкатегорий и статей связанных с удаляемой категорией. Если такие существуют, то пользователю сообщается об ошибке, иначе категория удаляется.

List of users

SQL terminal

Delete	Block	Unblock	Make admin	Disrank
	ID	Username	Email	Confirmed
<input type="checkbox"/>	1	username1 <i>(admin)</i>	yatskevichfyodor@gmail.com	✓
<input type="checkbox"/>	2	username2	email1@mail.com	✓
<input type="checkbox"/>	3	username3	email2@mail.com	✗
<input type="checkbox"/>	4	username4	email3@mail.com	✓

```
public interface UserRepository extends JpaRepository<User, Long> {  
    User findByEmailIgnoreCase(String email);  
    User findByUsernameIgnoreCase(String username);  
    void deleteById(Long id);  
  
    @Modifying  
    @Query("update User u set u.confirmed = 1 where u.username = ?1")  
    void confirm(String username);  
  
    @Modifying  
    @Query("update User u set u.blocked = 1 where u.id = ?1")  
    void block(Long id);  
  
    @Modifying  
    @Query("update User u set u.blocked = 0 where u.id = ?1")  
    void unblock(Long id);  
}
```

SQL-терминал

Enter SQL query

Query response

Send query

```
public String sqlRequest(String request) {
    StringBuilder response = new StringBuilder("");
    try {
        PreparedStatement ps = connect.prepareStatement(request);
        ResultSet rs;
        Pattern selectPattern = Pattern.compile("[sS][eE][lL][eE][cC][tT].*");
        Pattern describePattern = Pattern.compile("[dD][eE][sS][cC][rR][iI][bB][eE].*");
        Pattern insertPattern = Pattern.compile("[iI][nN][sS][eE][rR][tT].*");
        Pattern deletePattern = Pattern.compile("[dD][eE][lL][eE][tT][eE].*");
        Pattern updatePattern = Pattern.compile("[uU][pP][dD][aA][tT][eE].*");
        if (selectPattern.matcher(request).matches() || describePattern.matcher(request).matches()) {
            rs = ps.executeQuery();

            while (rs.next()) {
                for (int i = 0; i < rs.getMetaData().getColumnCount(); i++) {
                    response.append(rs.getObject("columnIndex: " + i + 1));
                    response.append(" ");
                }
                response.append("\n");
            }
        }
        else if (insertPattern.matcher(request).matches() || deletePattern.matcher(request).matches() ||
            updatePattern.matcher(request).matches()) {
            ps.executeUpdate(request);
            response.append("Request was executed");
        }
        else {
            response.append("Prohibited operation");
        }
    } catch (SQLException e) {
        response.append(e.toString());
    }

    return response.toString();
}
```

SQL-запросы

Данные из БД для таблиц user, article, comment, rating:

```
mysql> select id, username, timestamp from user;
```

id	username	timestamp
1	username1	2018-01-01 00:00:00
2	username2	2018-01-01 00:46:37
3	username3	2018-03-24 00:46:37
4	username4	2018-03-24 00:46:37

4 rows in set (0.00 sec)

```
mysql> select id, title, timestamp, popularity, author_id, category_id from article;
```

id	title	timestamp	popularity	author_id	category_id
1	Replicators	2018-03-09 00:00:00	1	1	3
2	Resident evil (film series)	2018-04-09 00:00:00	7	1	2
3	Spring Framework	2018-05-09 00:00:00	1	1	8
4	A perfect circle	2018-05-10 00:00:00	1	1	7
5	Angular	2018-05-11 00:00:00	1	1	9
6	Planet	2018-04-22 16:11:22	2	1	1
7	Forests and climate change	2018-05-13 17:34:38	5	2	10

7 rows in set (0.00 sec)

```
mysql> select * from comment;
```

id	text	timestamp	article_id	author_id
1	nice	2018-03-24 19:03:35	1	1
2	good	2018-03-24 19:03:40	1	1
3	Nice film	2018-05-13 17:59:37	2	2
4	I shoould try it	2018-05-13 18:00:02	5	2
5	should*	2018-05-13 18:00:12	5	2
6	cool spider	2018-05-13 18:00:48	1	2

6 rows in set (0.00 sec)

```
mysql> select * from rating;
```

value	article_id	user_id	timestamp
LIKE	1	2	2018-04-13 00:46:37
LIKE	2	1	2018-05-13 00:42:37
LIKE	2	2	2018-04-13 00:43:37
DISLIKE	2	4	2018-04-14 00:46:37
LIKE	5	2	2018-05-10 00:45:37

5 rows in set (0.00 sec)

Статистика: вывести количество статей, добавленных за последний месяц.

```
mysql> SELECT COUNT(*)
-> FROM article
-> WHERE timestamp BETWEEN (CURRENT_DATE - INTERVAL 1 MONTH) AND CURRENT_DATE
-> ;
+-----+
| COUNT(*) |
+-----+
|          5 |
+-----+
1 row in set (0.00 sec)
```

Having: вывести всех пользователей, количество статей которых больше 1.

```
mysql> SELECT
-> u.username,
-> COUNT(*) AS articles_number
-> FROM user u, article a
-> WHERE
-> u.id = a.author_id
-> GROUP BY author_id
-> HAVING articles_number > 1
-> ;
+-----+-----+
| username | articles_number |
+-----+-----+
| username1 |          6 |
+-----+-----+
1 row in set (0.00 sec)
```

Множественное сравнение: вывести всех пользователей из таблицы user, а также количество комментариев и количество статей для них с учетом того что их может и не быть.


```
mysql> SELECT
-> u.username,
-> c.comments_number,
-> a.articles_number
-> FROM
-> user u
-> LEFT JOIN (
-> SELECT
-> author_id,
-> COUNT(*) AS comments_number
-> FROM
-> comment
-> GROUP BY author_id
-> ) AS c ON u.id = c.author_id
-> LEFT JOIN (
-> SELECT
-> author_id,
-> COUNT(*) AS articles_number
-> FROM
-> article
-> GROUP BY author_id
-> ) AS a ON u.id = a.author_id
-> ;
```

username	comments_number	articles_number
username1	2	6
username2	4	1
username3	NULL	NULL
username4	NULL	NULL

4 rows in set (0.00 sec)

Множественное сравнение: вывести пользователей, количество статей которых больше количества комментариев, оставленных всеми другими пользователями за текущий месяц

```
mysql> SELECT u.username
-> FROM
->   user u
-> WHERE
->   (
->     SELECT COUNT(*)
->   FROM article a
->   WHERE
->     a.author_id = u.id
->   ) > (
->     SELECT COUNT(*)
->   FROM comment c
->   WHERE
->     c.author_id != u.id
->   AND c.timestamp BETWEEN DATE_FORMAT(NOW() , '%Y-%m-01') AND NOW()
->   )
-> ;
```

username
username1
username2

```
2 rows in set (0.00 sec)
```

Множественное сравнение: вывести информацию о статьях, рейтинг которых более N (лайков больше 1) был набран в течение месяца со дня публикации.

```
mysql> SELECT id, title, timestamp, popularity
-> FROM article a
-> WHERE
->   (
->     SELECT COUNT(*)
->   FROM rating r
->   WHERE
->     r.value = 'LIKE'
->     AND r.article_id = a.id
->   ) > 1
-> ;
```

id	title	timestamp	popularity
2	Resident evil (film series)	2018-04-09 00:00:00	7

```
1 row in set (0.00 sec)
```