**4.** РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

В данной главе более детально будет рассмотрена реализация некоторых функций и процессов разрабатываемого программного средства.

**4.1** Процесс создания задачи

Процесс создания задачи является одним из ключевых процессов, разрабатываемого программного средства. На рисунке 4.1 представлен алгоритм работы данного процесса в виде диаграммы активности.

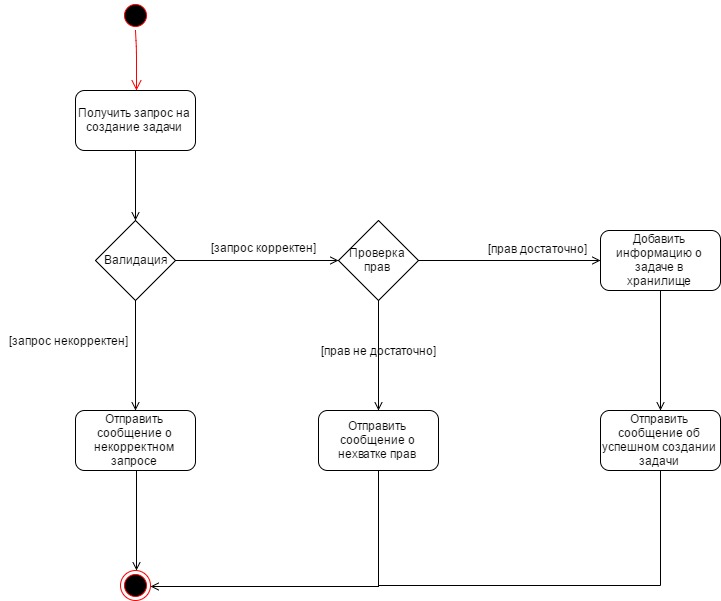


Рисунок 4.1 – Диаграмма активности процесса создания задачи

После получения запроса на создание задачи производится его валидация: у задачи должно присутствовать название, и его длина должна быть не больше восьмидесяти символов, у задачи должно быть описание не более 150 символов, у задачи должен быть указан проект, которому она принадлежит. Если валидация не была пройдена, на сторону клиента отправляется сообщение о некорректном запросе.

В случае успешного прохождения валидации выполняется проверка прав пользователя, чтобы выяснить может ли данный пользователь создавать задачи на данном проекте. В случае, если у пользователя недостаточно прав, ему отправляется сообщение об этом.

Если у пользователя оказалось достаточно прав для создания задачи, производится добавление задачи в базу данных, и отправка пользователю уведомления об успешном создании задачи.

**4.2** Поиск задачи

Процесс поиска задач необходим для отображения задач, удовлетворяющих некоторому критерию. Поиск может осуществляться по тегам или по названию задачи. Поиск задач производится в рамках одного проекта. На рисунке 4.2 представлена диаграмма активности для процесса поиска задач.

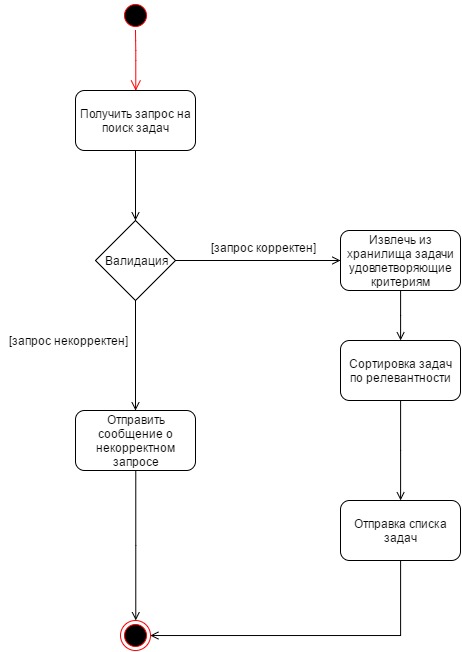


Рисунок 4.2 – Диаграмма активности процесса поиска задач

После получения запроса на поиск задач производится валидация запроса: строка запроса не должна быть пустой. Если вылидация не была пройдена, то пользователю отравляется уведомление о некорректно введенном запросе.

В случае успешного прохождения валидации из хранилища извлекаются все задачи, удовлетворяющие критериям поиска. Затем задачи сортируются по релевантности (по максимальному удовлетворению критериям запроса) и производится отправка списка задач клиенту. Ниже представлен SQL-запрос для поиска задач и метод уровня бизнес-логики searchTaks.

SELECT ta\_id, ta\_name, ta\_summary, ta\_description FROM task where ta\_name like :task\_name or   
ta\_id in (select tt\_task\_id from tag\_task inner join tag on lb\_id=tt\_tag\_id where lb\_name in (:tags)) and ta\_project\_id=:project\_id;

@Override  
public List<Task> searchTasks(Set<Tag> tags, String taskName, int projectId) {  
 List<Task> tasks = taskDao.searchTasks(tags, taskName, projectId);  
  
 for(Task task: tasks){  
 task.setTags(taskDao.getTaskTags(task.getId()));  
 }  
  
 Collections.sort(tasks, new RelevantTaskComparator(tags));  
  
 return tasks;  
}

**4.3** Организация доступа к базе данных

В разрабатываемом продукте доступ к базе данных осуществляется с помощью модуля spring-jdbc фреймворка Spring. Для непосредственного исполнения SQL-запросов используется класс NamedParameterJdbcTemplate.

При извлечении данных таким способом необходимо реализовать интерфейс RowMapper и параметризировать его типом возвращаемого объекта. Ниже приведен листинг, где реализован интерфейс RowMapper для извлечения проектов из базы данных.

private final RowMapper<Project> projectRowMapper = (resultSet, i)->{

Project project = new Project(); project.setId(resultSet.getInt(TableColumn.PROJECT\_ID)); project.setName(resultSet.getString(TableColumn.PROJECT\_NAME));

project.setUsersNumber(resultSet.getInt(TableColumn.USER\_COUNT));

project.setDescription(resultSet.getString(TableColumn.PROJECT\_DESCRIPTION));

project.setTotalTaskCount(resultSet.getInt(TableColumn.TASK\_COUNT));

User lead = new User();

lead.setUsername(resultSet.getString(TableColumn.USERNAME));

lead.setFirstName(resultSet.getString(TableColumn.FIRST\_NAME));

lead.setLastName(resultSet.getString(TableColumn.LAST\_NAME));

project.setLead(lead);

return project;

};

В листинге выше осуществляется извлечение информации о проекте вместе с его лидером посредством извлечения данных из экземпляра класса ResultSet.

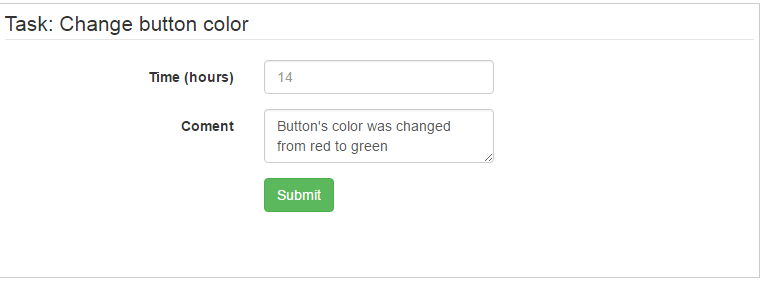
Для того чтобы выполнить запрос необходимо вызвать метод queryForList объекта класса NamedParameterJdbcTemplate и передать в него три параметра: SQL-запрос (параметры запроса должны быть отмечена как <:имя\_параметра>), одну из реализаций интерфейса SqlParameterSource (здесь отражается соответствие между именем параметра и его значением) и реализацию интерфейса RowMapper.

**4.4** Построение отчетов о затраченном времени

Данный модуль предоставляет возможность сотрудникам составлять отчет о времени, которое они затратили на решение той или иной задачи. При построении отчета пользователь должен ввести время в часах, которое он затратил и описать в комментарии какая работа была произведена. На рисунке 4.3 приведена форма, где пользователь может отправить отчет.

На рисунке 4.4 представлена диаграмма активности для процесса построения отчета о затраченном времени.

После получения запроса на построение отчета производится его валидация. Проверяется введенное количество часов (число должно быть положительным), а также проверяется наличие комментария к отчету. Если валидация не была пройдена, то клиенту отправляется сообщение об ошибке. В противном случае производится проверка прав пользователя.



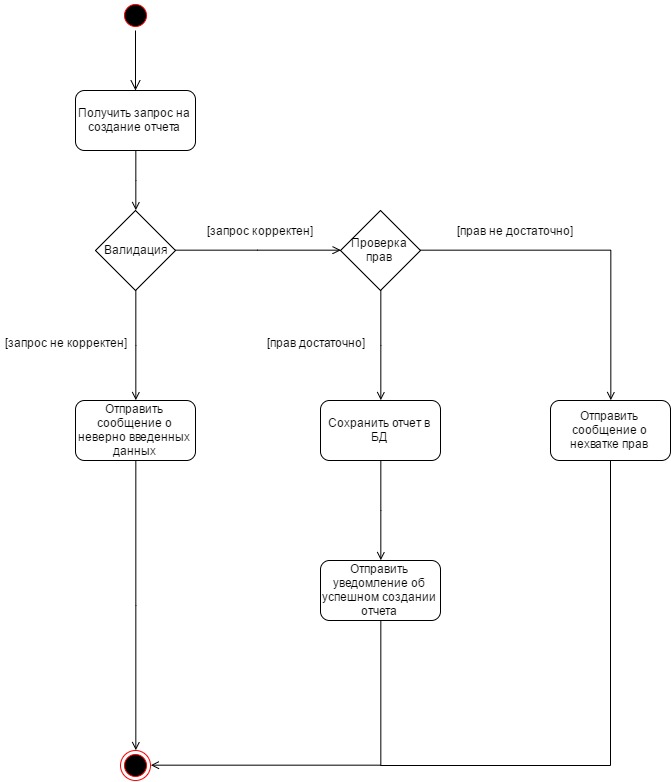
Рисунок 4.3 – Форма заполнения отчета о времени

Рисунок 4.4 – Диаграмма активности процесса построения отчетов

При проверке прав пользователя выяснятся, может ли данный пользователь списать свое рабочее время на данную задачу. Это необходимо для того, чтобы предотвратить списание рабочего времени на задачу, над которой пользователь не работал. Если у пользователя оказалось не достаточно прав, то ему отправляется уведомление об этом.

Если у пользователя достаточно прав, то производится сохранение отчета в базу данных, и отправка ему уведомления об успешном создании отчета.

**4.5** Изменение статуса задачи

Изменение статуса задачи является одной из ключевых операций в разрабатываемом программном продукте. Изменение статуса производится с помощью перетаскивания задач из одной колонки в другую. Каждый пользователь, работающий над проектом, имеет определенный набор прав, определяющий из какого статуса в какой задача может быть перемещена. На рисунке 4.5 изображена диаграмма активности для данного процесса.

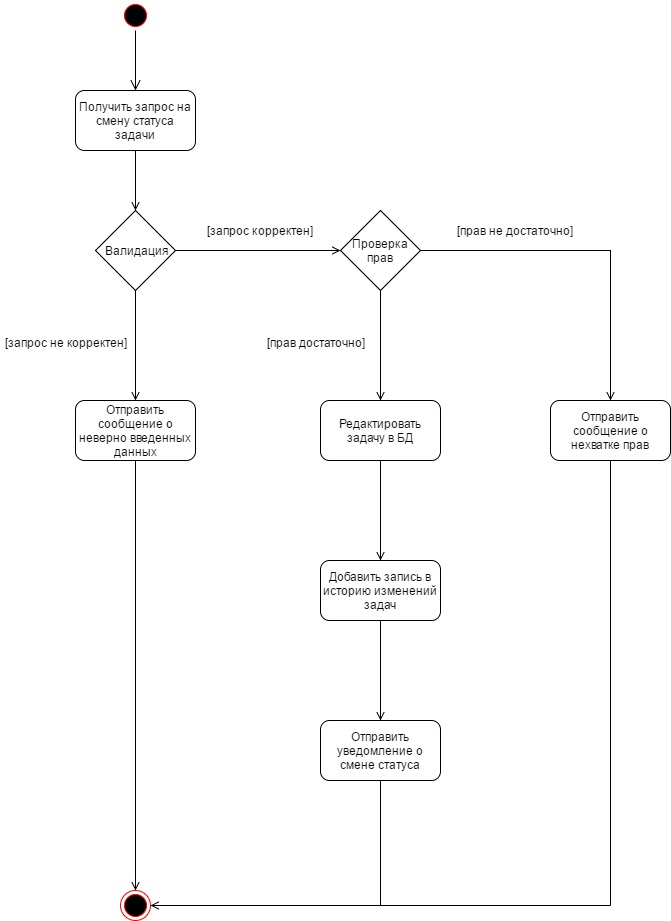


Рисунок 4.5 – Диаграмма активности процесса смены статуса задачи

После получения запроса на смену статуса задачи производится его валидация: у задачи должен быть указан ее идентификатор, новый и прежний статусы. В случае успешного прохождения валидации выполняется проверка прав пользователя, тем самым выясняется может ли он изменить статус задачи с текущего на новый. Если прав у пользователя достаточно, то задача обновляется в базе данных, добавляется запись в историю изменения задач, а пользователю приходит уведомление о смене статуса задачи. Ниже приведен фрагмент кода, обновляющий статус задачи.

@Override

@Transactional

@PreAuthorize("@permissionDao.canChangeStatus(principal.username, task.taskStatus, newStatus)")

public void changeTaskStatus(Task task, TaskStatus newTaskStatus){

taskDao.updateStatus(task, newTaskStatus);

HistoryItem historyItem = new HistoryItem(task, newTaskStatus);

taskHistoryDao.createHistory(historyItem);

}

Аннотация @Transactional говорит о том, что метод changeTaskStatus выполняется в рамках одной транзакции. Аннотация @PreAuthorize проверяет обладает ли пользователь достаточными правами для изменения статуса задачи. Метод updateStatus обновляет статус задачи в базе данных, а метод createHistory добавляет запись в историю изменения задач.

**4.6** Добавление пользователя в проектную группу

Каждый созданный проект обладает набором проектных групп, в которые могут быть добавлены пользователи с одинаковой ролью (например, тестировщики, разработчики и др.). Каждая группа обладает своим списком прав, которые позволяют пользователям совершать те или иные действия. В данном подразделе описывается процесс добавления пользователя в проектную группу (см. рис. 4.6).

После получения запроса на создание проектной группы выполняется проверка прав пользователя, тем самым выясняется может ли данный пользователь добавлять участников в группу (добавление в группу может осуществить только руководитель проекта). Если прав достаточно, то производится редактирование списка пользователей, входящих в группу и отправляется уведомление об успешном добавлении пользователя. В обратном случае отправляется уведомление о нехватке прав.

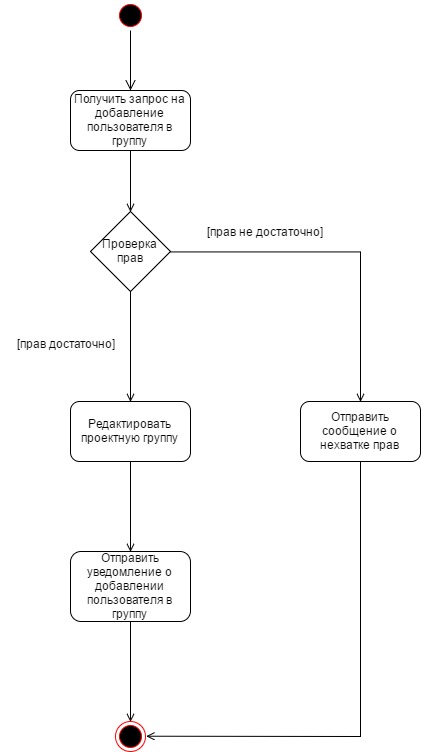


Рисунок 4.6 – Диаграмма активности процесса добавления пользователя в группу

Ниже приведен фрагмент кода, добавляющий пользователя к проектной группе.

Аннотация @Transactional говорит о том, что метод addUserToGroup выполняется в рамках одной транзакции. Аннотация @PreAuthorize проверяет обладает ли пользователь достаточными правами для изменения статуса задачи. Метод addToGroup добавляет пользователя в базу данных.

@Override

@Transactional

@PreAuthorize("@projectDao.getProjectLead(projectId).username == principal.username")

public void addUserToGroup(String username, int projectId, int groupId){

groupDao.addToGroup(username, groupId);

}

Аннотация @Transactional говорит о том, что метод addUserToGroup выполняется в рамках одной транзакции. Аннотация @PreAuthorize проверяет обладает ли пользователь достаточными правами для изменения статуса задачи. Метод addToGroup добавляет пользователя в базу данных.