Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет: «Информационных технологий и прикладной математики»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Базы данных»

**Курсовой проект**

**Тема: «Каталог автомобилей»**

|  |  |
| --- | --- |
| Cтуденты: | Ирисов Т. А. |
| Группа: | М8О-309Б-22 |
| Преподаватель: | Киндинова В. В. |
| Дата: |  |
| Оценка: |  |

Москва 2024

Оглавление

Таблицы и схемы данных 3

Таблица users 3

Таблица tasks 3

Таблица user\_task 4

Таблица dorms 4

Таблица blocks 4

Таблица user\_block 4

Таблица dorm\_block 5

Таблица history 5

Таблица requests 5

Взаимосвязи 6

Схема приложения 7

Сервисы и модели. 9

User 9

Tasks 12

History 14

Dorm 16

Request 18

Список используемых источников. 22

Репозиторий проекта 23

# 

# Таблицы и схемы данных

В данной работе БД состоит из девяти таблиц: users, tasks, user\_task, blocks, user\_block, dorms, dorm\_block, history, requestst.

## Таблица users

Описание: Хранит информацию о пользователях системы.

### Поля:

- user\_id: уникальный идентификатор пользователя (первичный ключ).

- user\_login: логин пользователя.

- user\_password: пароль пользователя.

- user\_name: имя пользователя.

- user\_surname: фамилия пользователя.

- user\_middle\_name: отчество пользователя (опционально).

- user\_role: роль пользователя, по умолчанию — 'user'.

## Таблица tasks

Описание: Хранит информацию о задачах.

### Поля:

- task\_id: уникальный идентификатор задачи (первичный ключ).

- task\_name: название задачи.

- task\_type: тип задачи, по умолчанию — 'single' (одноразовая задача).

## Таблица user\_task

Описание: Связывает пользователей и задачи, определяя, какие задачи назначены пользователям.

### Поля:

- user\_id: идентификатор пользователя (внешний ключ на users.user\_id).

- task\_id: идентификатор задачи (внешний ключ на tasks.task\_id).

## Таблица dorms

Описание: Хранит информацию об общежитиях.

### Поля:

- dorm\_id: уникальный идентификатор общежития (первичный ключ).

- dorm\_name: название общежития.

- dorm\_adress: адрес общежития.

## Таблица blocks

Описание: Хранит информацию о блоках в общежитиях.

### Поля:

- block\_id: уникальный идентификатор блока (первичный ключ).

- block\_number: номер блока.

## Таблица user\_block

Описание: Связывает пользователей и блоки, определяя, в каком блоке проживает пользователь.

### Поля:

- user\_id: идентификатор пользователя (внешний ключ на users.user\_id и первичный ключ).

- block\_id: идентификатор блока (внешний ключ на blocks.block\_id).

## Таблица dorm\_block

Описание: Связывает блоки и общежития, определяя, в каком общежитии находится блок.

### Поля:

- block\_id: идентификатор блока (внешний ключ на blocks.block\_id).

- dorm\_id: идентификатор общежития (внешний ключ на dorms.dorm\_id).

## Таблица history

Описание: Хранит историю действий пользователей в системе.

### Поля:

- id: уникальный идентификатор записи (первичный ключ).

- dorm\_id: идентификатор общежития, связанного с действием (внешний ключ на dorms.dorm\_id).

- block\_id: идентификатор блока, связанного с действием (внешний ключ на blocks.block\_id).

- user\_id: идентификатор пользователя, совершившего действие (внешний ключ на user\_block.user\_id).

- action: описание действия.

- date\_time: время и дата действия, по умолчанию — текущее время.

## Таблица requests

Описание: Хранит информацию о заявках, созданных пользователями.

### Поля:

- request\_id: уникальный идентификатор заявки (первичный ключ).

- dorm\_id: идентификатор общежития, связанного с заявкой (внешний ключ на dorms.dorm\_id).

- block\_id: идентификатор блока, связанного с заявкой (внешний ключ на blocks.block\_id).

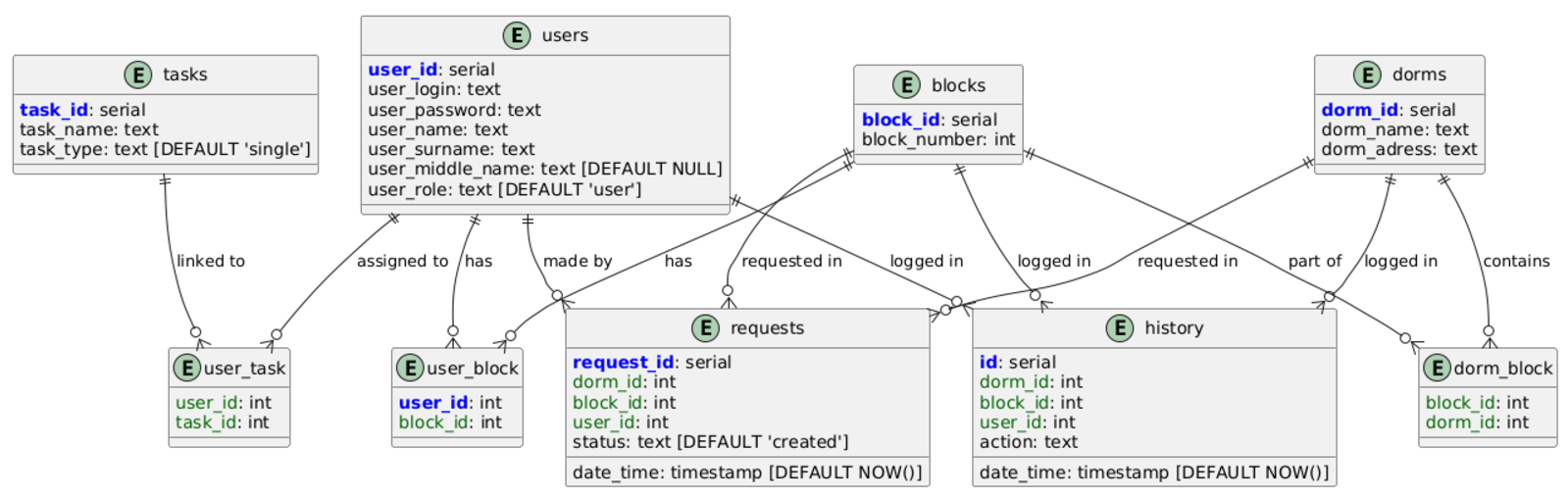
- user\_id: идентификатор пользователя, создавшего заявку (внешний ключ на users.user\_id).

- status: статус заявки, по умолчанию — 'created'.

- date\_time: время и дата создания заявки, по умолчанию — текущее время.

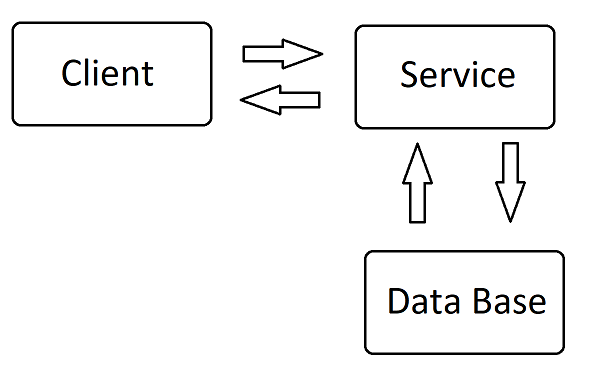
## Взаимосвязи

Таблицы используют внешние ключи для обеспечения связности данных между пользователями, задачами, общежитиями, блоками и историями действий. Основная структура построена так, чтобы каждая запись была четко привязана к конкретному пользователю, задаче или объекту (общежитию/блоку).



# 

# Схема приложения



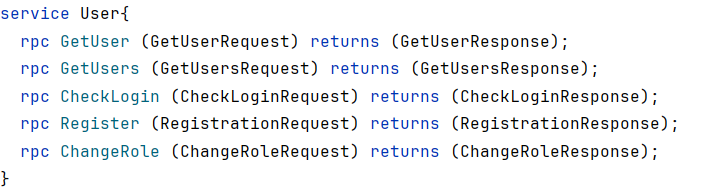
Данное приложение является многопользовательским и имеет сервисную структуру, то есть обращение клиента посылается не напрямую в базу данных, а на определенный сервис который находится на сервере с backend и уже этот сервис в зависимости от типа запрос посылает разные запросы в базу данных, и возвращает ответы клиенту. Такой подход является эффективным, легко масштабируемым, и безопасным, так как клиент не хранит в себе какую либо секретную информацию. Также хочу упомянуть что пароли в базе данных хранятся в хэшированном виде, что тоже является признаком безопасности.

# 

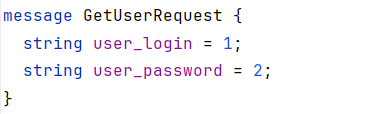
# Сервисы и модели.

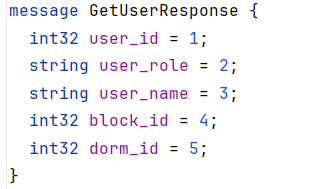
## User

Данный сервис работает с моделью User и имеет методы:



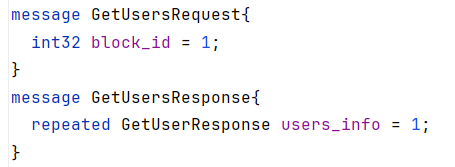
1. GetUser принимает на вход GetUserRequest:

  
  
и возвращает GetUserResponse:

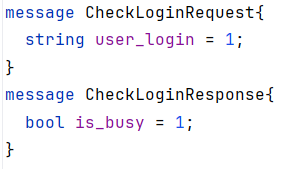


Данный сервис используется для авторизации в приложении.

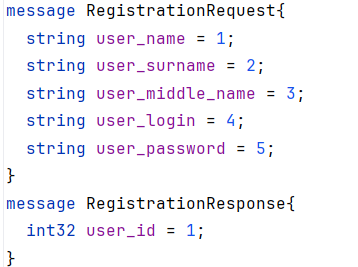
1. GetUsers принимает запрос GetUsersRequest и возвращает GetUsersResponse. Используется в приложении для просмотри информации и всех пользователях которые находятся с текущем блоке общежития



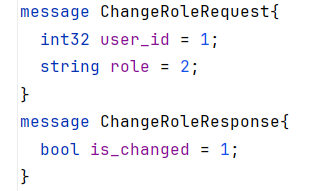
1. CheckLogin используется при регистрации для проверки занятости логина.



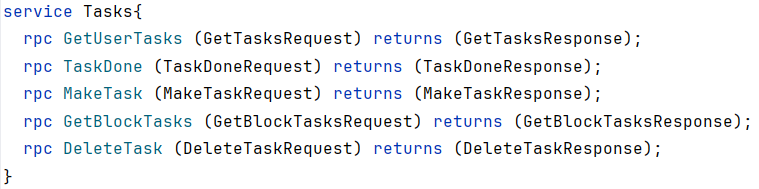
4) Register регистрирует нового пользователя после проверки логина и пароля на соответствие требованиям сложности и доступности.



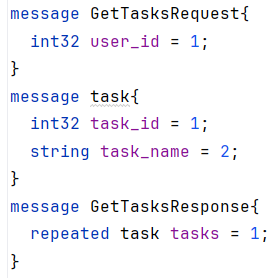
1. ChangeRole используется для назначения новых прав пользователю администратором, как назначение администратором, так и отбором прав администратора.



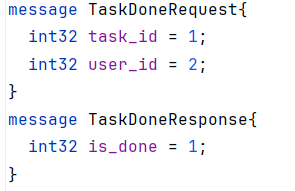
## Tasks



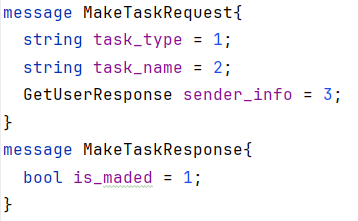
1. GetUserTasks используется после успешной авторизации, для получения всех актуальных задач для текущего пользователя. Все задачи представлены в виде списка в главном окне приложения для пользователя.



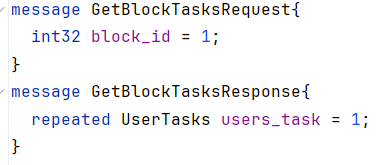
1. TaskDone. Данный метод вызывается при нажатии кнопки «Выполенено» на определенном задании, после чего данное задание будет перенесено из актуальных заданий в историю в базе данных.



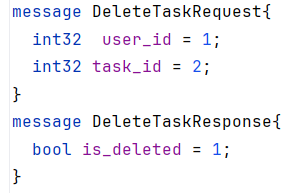
1. MakeTask. Вызывается при создании задач для участников блока в общежитии. Задачи могут быть как и одноразовыми, так и поочередными, то есть если один участник выполнил задание, то оно переходит следующему после вызова TaskDone.



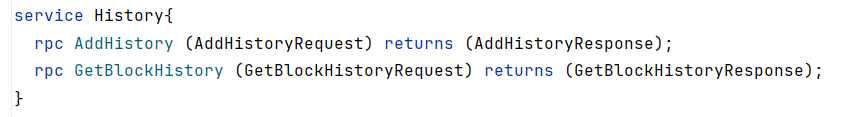
1. GetBlockTasks. Метод используется чтобы получить список актуальных заданий для всех участников блока. Данный метод доступен для всех типов участников.



1. DeleteTask. Метод доступный только администратору, который может удалить задание для определенного пользователя.

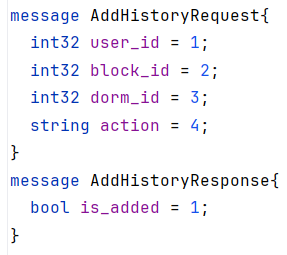


## History



1. AddHistory. Добавляет в историю какое либо значимое действие участника. Например если участник: выполнил задание, покинул блок,

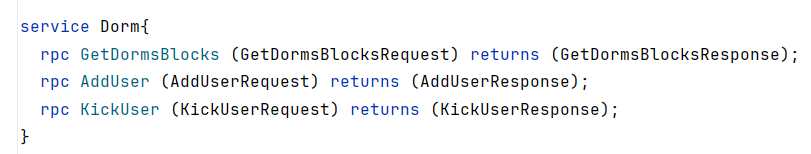
вступил в блок и тд.



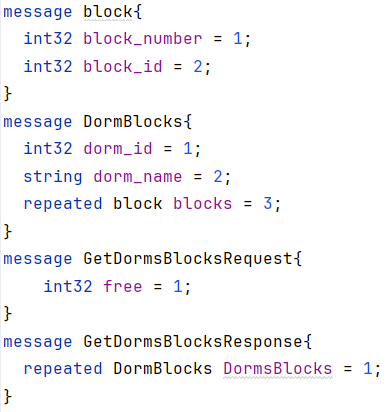
1. GetBlockHistory. Возвращает список всех значимых действий все участников блока, который может посмотреть любой участник блока.



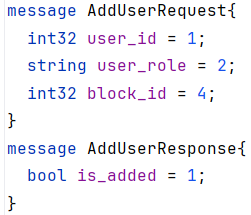
## Dorm



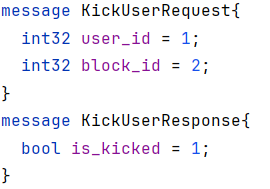
1. GetDormsBlocks. После регистрации пользователь может отправить заявку на вступления уже в сформированный блок или же создать свой из свободных. Метод возвращает список общежитий вместе с блоками.



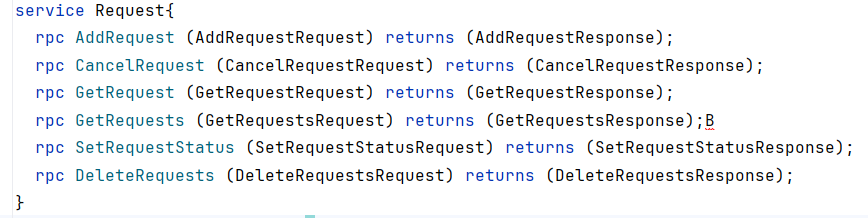
1. AddUser. Если пользователь отправил заявку на вступление в блок, администратор может одобрить вступления или отклонить. Если администратор одобряет вступление нового участника, то вызывается данный метод и добавляет участника в блок.



1. KickUser. Вызывается для удаления пользователя из блока. Данный метод доступен только администратору.

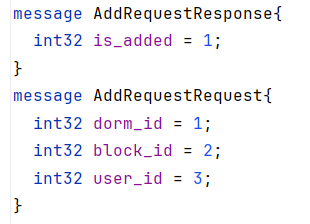


## Request

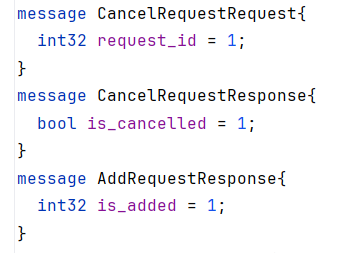


1. AddRequest. Вызывается после отправки заявки на вступление в блок.

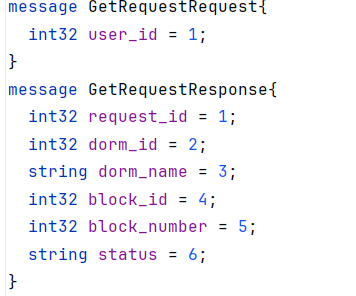
Администраторы получают данную заявку в окне с заданиями и могу ее принять или отклонить.



1. СancelRequest. Если администратор отклоняет заявку на вступление в блок, вызывается данный метод.



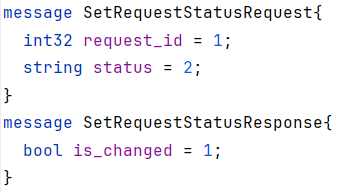
1. GetRequest. Используется для проверки наличия у пользователя каких либо запросов при входе в приложение, если же запросы у пользователя есть, то приложение сообщает пользователю об этом.



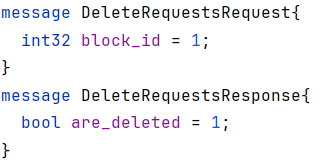
1. GetRequests. Используется для администраторов при входе в главное окно и проверяет на наличие каких либо запросов в текущий блок.



1. SetRequestStatus. Меняет текущий статут заявки в зависимости от ответа администратора. Заявка может быть принята или отклонена.



1. DeleteRequests. Метод удаляет все запросы блока, вызывается если последний участник блока покидает его. Участник ,который подал заявку в этот блок, может создать его сам и стать его администратором.



# Список используемых источников.

1. [«Запускаем PostgreSQL в Docker»](https://habr.com/ru/articles/578744/)
2. [«Методы хэширования паролей»](https://habr.com/ru/articles/765524/)
3. [«PsyCopg2 Docs»](https://www.psycopg.org/docs/)
4. [«GRPC python»](https://grpc.io/docs/languages/python/)
5. [«Qt Designer Manual»](https://doc.qt.io/qt-6/qtdesigner-manual.html)
6. [«PyQt5 Guide»](https://www.riverbankcomputing.com/static/Docs/PyQt5/)

# [Репозиторий проекта](https://github.com/TimeB30/db_project)

