# Предупреждение

Я впервые занимаюсь разработкой подобных систем (а в моем понимании, написание собственной клиент-серверной архитектуры, при том, что ей будут пользоваться, действительно является системой), поэтому может быть много калового кода, нестандартных архитектурных решений и куда же нам без hard кода. Своей целью я ставил прежде всего удобство использования, юзабельность, а также расширяемость. Данный документ является не только документацией моего кода и всего вокруг, но и *дневником разработчика*, который, в большей степени, я пишу для себя.

# База

Доступ в систему имеют люди (зарегистрированные) с одним из трех уровней доступа: Пользователь, Специалист и Администратор. Уровень доступа Пользователь имеет каждый зарегистрированный пользователь. Повысить статус пользователю может только Администратор, как и дать ключ для регистрации.

# API

Итак, чтобы пользоваться системой и не испытывать при этом cringe, необходимо быть авторизованным пользователем. Сделать это можно двумя способами:

1. Через логин и пароль;
2. Через token уже авторизованного пользователя.

Как вы понимаете, чтобы получить token, необходимо выполнить обычную авторизацию по логину и паролю. Для этого мы создаем instance класса UserPassport и в качестве аргументов передаем логин и пароль. В качестве ответа будет возвращен token авторизации.

Token авторизации – единственное, с чем доступна дальнейшая работа с сервером, поскольку передается со всеми следующими пакетами (как hash)

После входа в систему, пользователя перекидывает в основное окно, в котором по полученному токену сверяется с данными на сервере и предоставляет определенный уровень доступа к функционалу.