Использовано:

На серверной стороне:

* Express – Фреймворк для работы с HTTP, для создания апи
* База данных – PostgreSQL
* Sequelize – ORM для работы с бд
* Cors – Для того, чтобы можно было с браузера обращаться к серверу
* Dotenv – для переменных окружения
* Nodemon – Перезапускает автоматически сервер при изменениях

На клиентской части:

* React – Фреймворк для созданию веб приложения
* Node-sass – пакет для работы с препроцессором sass
* Redux – Менеджер глобального хранилища
* React-router-dom – для работы с переходами по страницам
* Axios – для запросов на сервер

Структура проекта

Главная папка/ - Хранится клиентская и серверная часть

Главная папка/client – Клиентская часть приложения

/client/src – Папка, где находится все разработка клиентской части приложения

Главная папка/server – Серверная часть приложения

/server/index.js – Центральный файл для всех настроек и запуска сервера

/server/db.js – Файл для создания бд

/server/.env – файл для глобальных переменных

/server/models – Папка для создания таблиц бд

/server/routes – Папка с файлами для работы роутов(маршрутизация сервера)

/server/controllers – Папка с классами по работе веб приложения по определённым роутам(путям)

**Роуты к серверу(пути для обращения к серверу, маршрутизация сервера)**

**/user**

Метод – post, путь /registration – Отправка данных на регистрацию пользователя

Метод – post, путь /login – Отправка данных на вход в аккаунт

Метод – get, путь /auth – Проверка авторизован ли пользователь

Метод – delete, путь /add – Пополнение баланса

**/operation**

Метод – get, путь /:id – Получение операций пользователя

Метод – post, путь / – Создание перевода между пользователями

Метод – get, путь /coming/:id. – Получение суммы операций

Метод – get, путь /consum/:id. – Получение суммы расходов

**Классы**

class UserController класс по авторизацией

Его методы:

* registartion – Регистрация пользователя
* login – Вход в аккаунт
* check – Проверка авторизован ли пользователь
* addBalance – Пополнение баланса пользователю

class OperationController класс по работе со странами

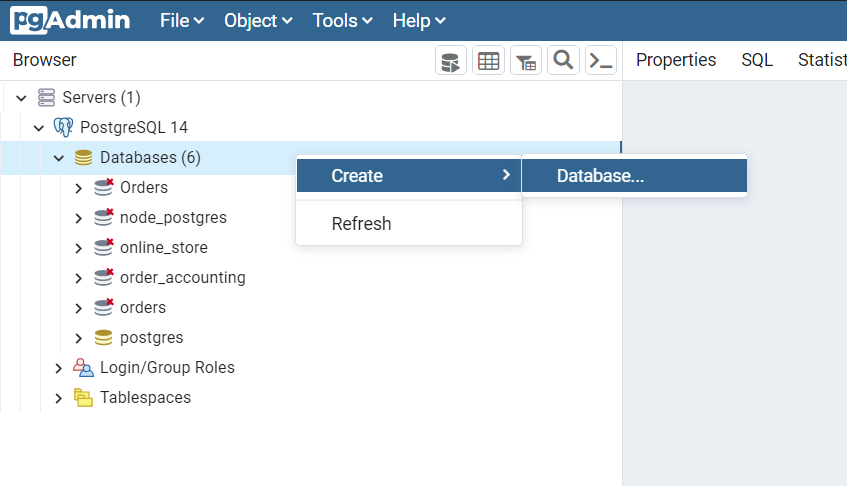
Его методы:

* send – Перевод средств пользователю
* getAll – Получение операций пользователя
* comingBalance – Получение суммы положительных операций
* consumBalance – получение суммы расходных операций

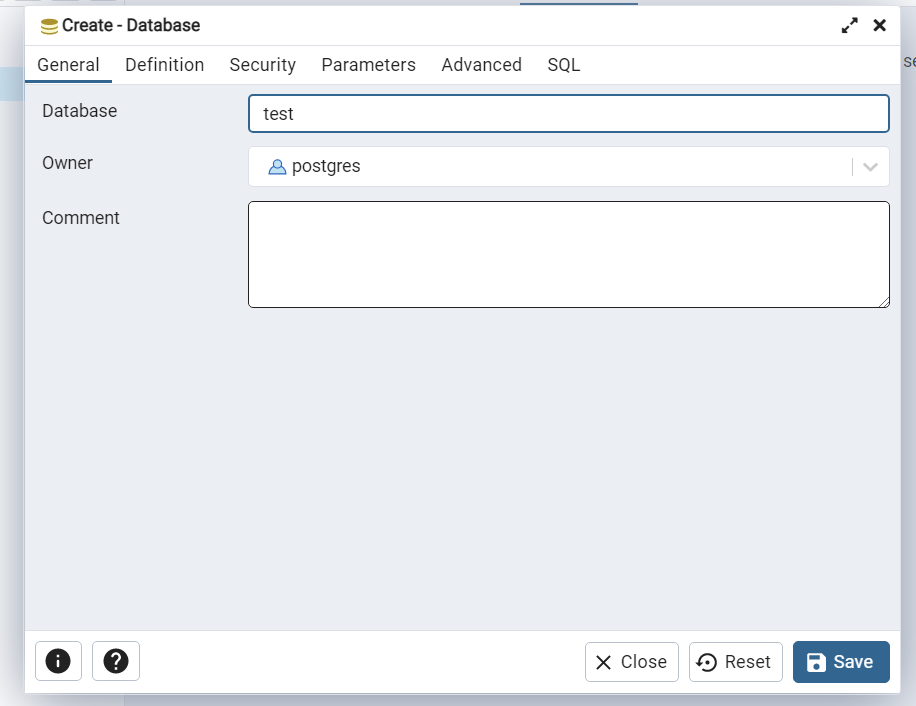
**Запуск проекта**

Для начала нужно скачать pgAdmin, при установке данного ПО надо запомнить данные user(по умолчанию - postgres), port(по умолчанию - 5432), **password**.

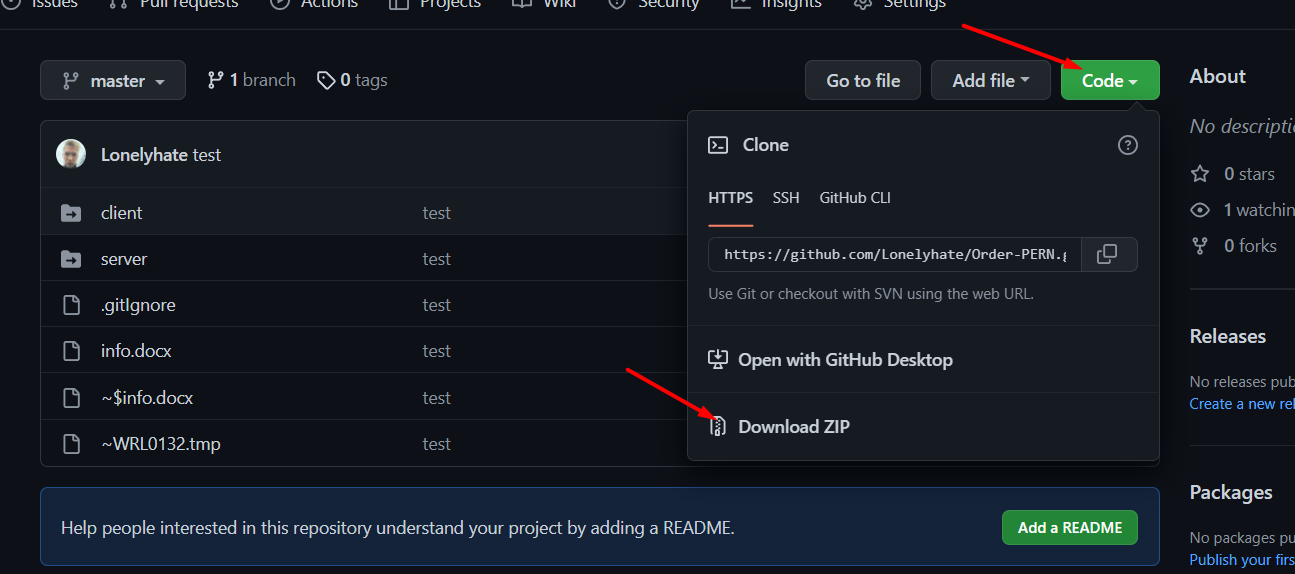
После установки нужно создать бд для сервера



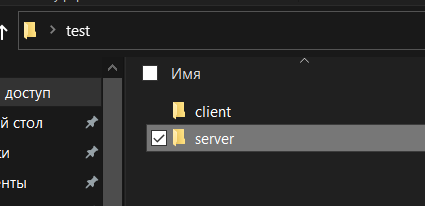
Создаем бд



В репозитории скачиваем архив

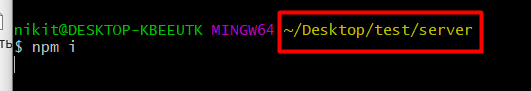


Создаем папку для проекта и переносим туда папку сервер и клиент



Заходим в папку сервер

Открываем консоль и пишем команду npm i



После успешной установки зависимостей точно также проделываем и в папке client



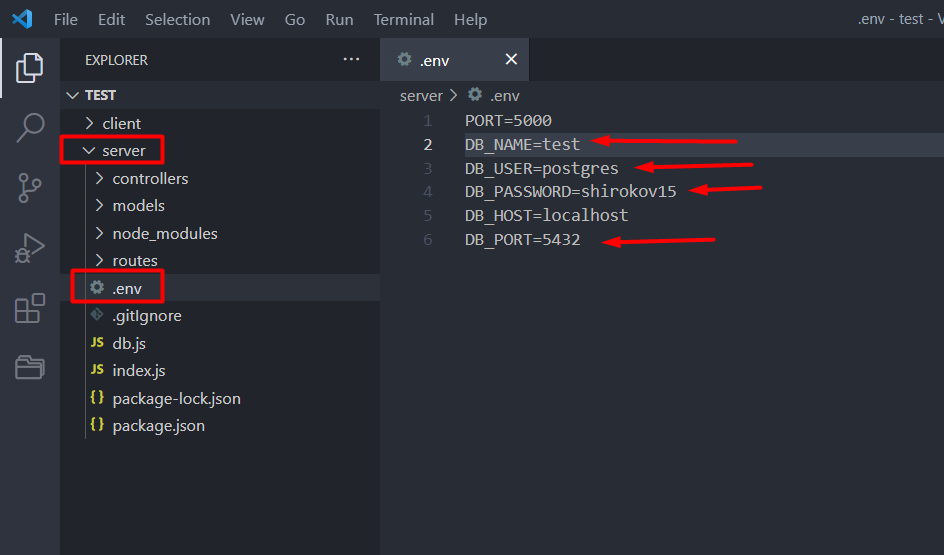
Далее открываем данный проект с помощью редактора какого нибудь и переходим в файл .env для подключение бд

DB\_NAME = [Вписать имя бд, которое вы создали]

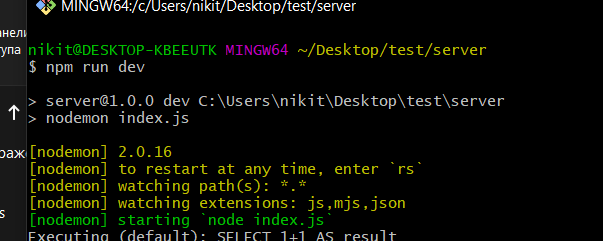
DB\_USER = [По идеи можно не исправлять]

DB\_PASSWORD = [Пароль при установке ПО pgAdmin]

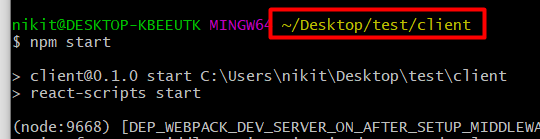
DB\_PORT = [По умолчанию должен был быть 5432, поэтому менять не надо]



Далее открываем консоль в папке сервер и прописываем npm run server. Не закрываем данную консоль, пока находитесь на сайте, благодаря ней работает сервер



Теперь запускаем клиент, точно также заходим в папку client и открываем в ней консоль и прописываем в ней npm start, точно также не закрываем консоль, потому что она работу осуществляет клиента. Данная команда автоматически открывает страницу в браузере с приложением



Все, приложение запущено