# Опис серверної частини.

Серверна частина нашого додатку побудована на Node.js. [Node.js](https://nodejs.org/uk/" \o "https://nodejs.org/uk/)[®](https://nodejs.org/uk/" \o "https://nodejs.org/uk/) — це JavaScript–оточення побудоване на JavaScript–рушієві Chrome V8.

У документації до Node.js зазначено, що як асинхронне подієве JavaScript–оточення, Node.js спроектований для побудови масштабованих мережевих додатків. Для кожного з’єднання викликається функція зворотнього виклику, проте коли з’єднань немає Node.js засинає.

Це контрастує з більш загальною моделлю в якій використовуються паралельні OS потоки. Такий підхід є відносно неефективним та дуже важким у використанні. Більше того, користувачі Node.js можуть не турбуватись про блокування процесів, оскільки немає жодних блокувань. Майже жодна з функцій у Node.js не працює напряму з I/O, тому процес не блокується ніколи. Оскільки нічого не блокується на Node.js легко розробляти масштабовані системи.

Для забезпечення створення http-сервера і роботи додатка за протоколом http був використаний фреймворк [Express - Fast, unopinionated, minimalist web framework for Node.js](https://expressjs.com). Як стверджує сайт Express.js Express — це мінімальний і гнучкий фреймворк веб-додатків Node.js, який надає надійний набір функцій для веб- та мобільних додатків.

Для взаємодії з клієнтською частиною нашого додатку за протоколом Web Socket була використана бібліотека [Socket.IO](https://socket.io/docs/v4/). Socket.IO — це бібліотека, яка забезпечує двонаправлений зв'язок між клієнтом і сервером із низькими затримками та на основі подій. Вона побудована на основі протоколу WebSocket і надає додаткові гарантії, такі як повернення до тривалого опитування HTTP, якщо браузер не підтримує роботу за протоколом web socket або автоматичне повторне підключення.

Для зберігання даних додатку використовується система керування базою даних [PostgreSQL](https://www.postgresql.org). PostgreSQL – це потужна система об’єктно-реляційних баз даних з відкритим вихідним кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які безпечно зберігають і масштабують найскладніші робочі навантаження даних.

Проаналізувавши предметну область нашого завдання прийшли до висновку, у нас існує три окремих сутності, а саме користувач, «кімната для переговорів» або чат, повідомлення.

Користувач має наступні властивості: логін, пароль, ім’я, що відображається, аватар – графічне зображення, яке пов’язане з цим користувачем. Для зберігання аватару будемо використовувати файлову систему, а в базі даних будемо зберігати тільки ім’я файлу.

«Кімната для переговорів» повинна мати такі властивості

Ідентифікатор, тип кімнати (коли програма буде розвиватися, з’явиться можливість створювати не тільки персональні чати, але і групи та інформаційні канали. Крім того в подальшому функціонал програми буде передбачати можливість блокування «кімнат» адміністратором або співрозмовником. Тому в базі даних такі можливості повинні бути передбачені. Назва кімнати є індивідуальною для кожного користувача і тому не повинна бути пов’язана з цією таблицею.

Повідомлення має такі властивості: автор повідомлення, кімната, в яку це повідомлення відправлено, та власне текст повідомлення.

Сутності мають наступні зв’язки;

Користувачі пов’язані з повідомленнями зв’язком «один до багатьох» через поле в таблиці повідомлень – «автор».

«Кімнати» пов’язані з повідомленнями зв’язком «один до багатьох» через поле в таблиці повідомлень «призначення».

«Кімнати» пов’язані з користувачами зв’язком «багато до багатьох». Для реалізації такого зв’язку необхідно ввести додаткову таблицю, що пов’язує кімнати та користувачів. Додатково в базі даних заведена таблиця длч зберігання сесій фреймворка Express. ERD діаграма наведена нижче.

