Основні сутності проекту

- **Користувачі**: Профілі, налаштування, ролі(бренди, креатори і адміни, з різними рівнями доступу та функціональністю).
- **Кампанії**: весь опис та деталі створених брендами кампаній.
- **Заявки**: Подання та управління заявками креаторів на кампанії (статуси, коментарі і тд).
- **Месенджер**: Функціонал для взаємодії між брендами і креаторами.
- **Аналітика**: Збір та візуалізація даних про кампанії та участь у них

Тех стек

- **NestJS (Node.js)** – фреймворк
- **Temporal** – довготривалі бізнес-процеси (воркфлови заявок і кампаній), автоматичні retry, вся історія процесів, reset зафейлених активіті
- **AWS Cognito** – готове рішення для автентифікації, авторизації, управління ролями
- **PostgreSQL + TypeOrm** – основна база даних
- **Apache Kafka** – брокер повідомлень та подій між сервісами, сервіс аналітики, нотифікацій
- **Redis** – швидкодія, кешування
- **Docker + Docker Compose** – ізоляція та деплой
- **Prometheus + Grafana** – моніторинг ресурсів і базових метрик

Авторизація та розмежування доступів

Для авторизації та розмежування доступів використав би **AWS Cognito**:

- SSO, MFA, email/social логін
- Визначення ролей (бренди, креатори, адміни) через групи користувачів в Cognito.
- JWT-токени для автентифікації і авторизації на бекенді
- Використання атрибутів користувачів для розмежування доступів до ресурсів та функціоналу

Майбутнє розширення

- **Месенджер** – WebSockets + kafka events
- **Аналітика** – окремий сервіс, що підписаний на Kafka-події (ClickHouse)
- **Заявки** – всі обробки заявок через Temporal task queue для надійності, збереження історії та більшої гнучкості
- **Нотифікації** – email/sms /push як окремий мікросервіс

Стабільність та масштабованість системи:

- **Мікросервісна архітектура та ізоляція через Dokcer**: Розділення функціоналу на мікросервіси для зменшення впливу помилок і зручного масштабування.
- **Кешування**: Використання Redis для швидкодії та зменшення навантаження на базу даних.
- **Резервне копіювання даних:** PostgreSQL та Redis для запобігання втрат даних**.**
- **Горизонтальне масштабування сервісів**: Kubernetes для автоматизації деплою та оркестрації контейнерів, а також Temporal workers в залежності від кількості Task Queues та кількості задач в черзі/чергах