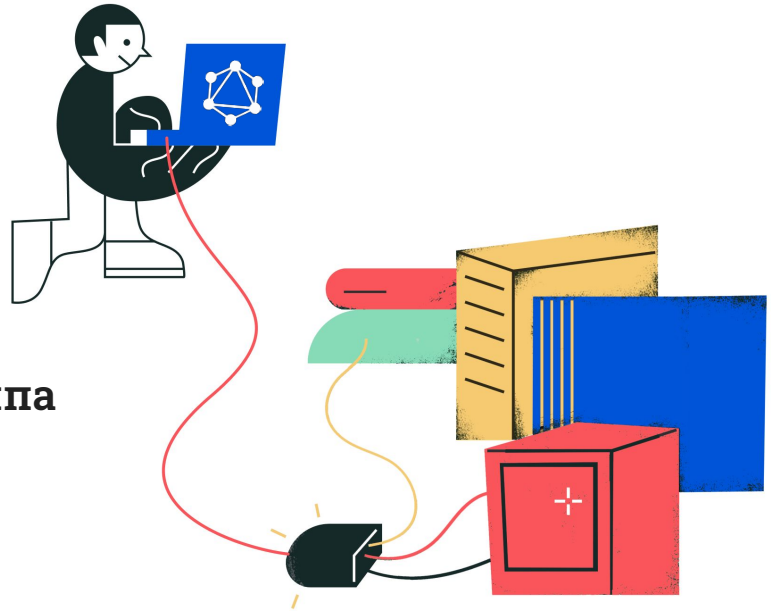
GraphQL на клиенте и на сервере



От идеи до прототипа

CSSSR

Цели воркшопа

- Сформировать представление о веб-приложении как о цельном продукте, включающем в себя работу на клиенте, сервере, взаимодействие с БД и т.д.
- Рассказать о современных инструментах, которые могут помочь и упростить процесс создания полноценного веб-приложения
- Продемонстрировать внутреннее устройство GraphQL, как основной технологии, определяющей архитектуру приложения. Рассмотреть его отличия от существующих подходов, показать преимущества и кейсы использования

План воркшопа

1. **Общая схема приложения. Демо**

2. **Создание клиента**
   a. Next.js, SSR
   b. Роутинг
   c. Компоненты

3. **GraphQL**
   a. Отличие от REST
   b. Преимущества и недостатки

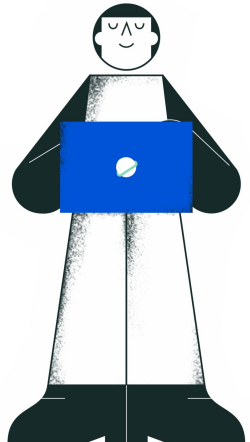
4. **Создание сервера**
   a. Prisma
   b. БД, миграции
   c. Построение схемы

5. **Получение данных**
   a. GraphQL Playground
   b. Apollo
   c. Обработка запросов, кэш

6. **Улучшаем клиент**
   a. Интроспекция
   b. Кодогенерация
   c. Фрагменты

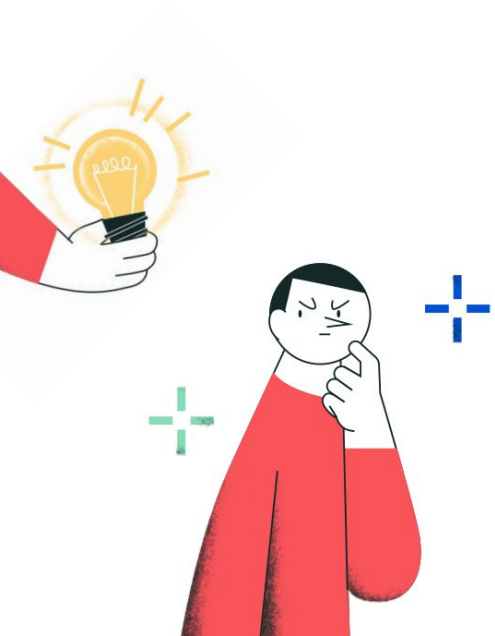
7. **Прокачиваем сервер**
   a. Мутации
   b. Авторизация и аутентификация

8. **Запуск!**



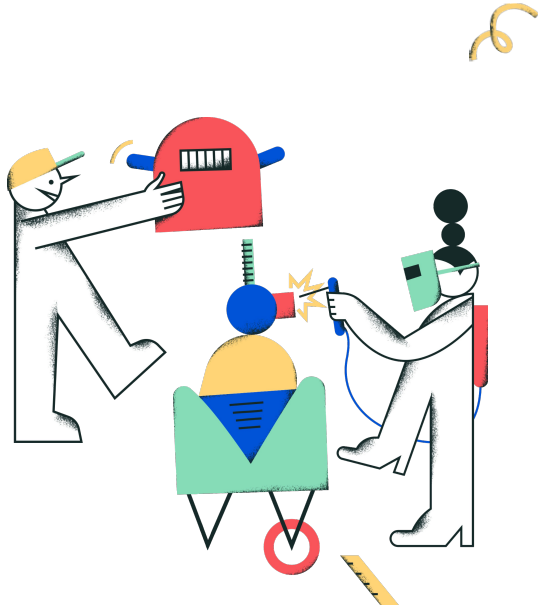
CSSSR

1. Общая схема приложения. Демо



2. Создание клиента

1. Next.js
2. Концепция SSR
3. Сравнение SSR и CSR
4. Роутинг. Файловая структура
5. Создание компонентов
6. Стилизация в Next.js
7. Material-UI
8. Концепция CSS-in-JS
9. CSS и SSR
10. _app и _document
11. Навигация



Next.js

Что: Фреймворк на основе React. Предоставляет “из коробки” роутинг, SSR, Static Exporting, Code Splitting, CSS-in-JS

Плюсы: Хорошая поддержка, крутая документация, zero configuration

Минусы: Сложность изоморфных приложений, проблемы с поддержкой CSS

Аналоги: CRA, Gatsby, Hexo, Angular

Material-UI

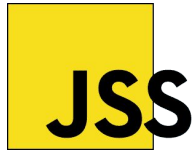
Что: Одна из самых популярных библиотек компонентов

Плюсы: Очень просто интегрировать, гибкая кастомизация, поддержка Typescript и SSR

Минусы: Малое количество компонентов, больше подходит для мобильных интерфейсов

Аналоги: Ant Design, Bootstrap, PrimeReact

JSS



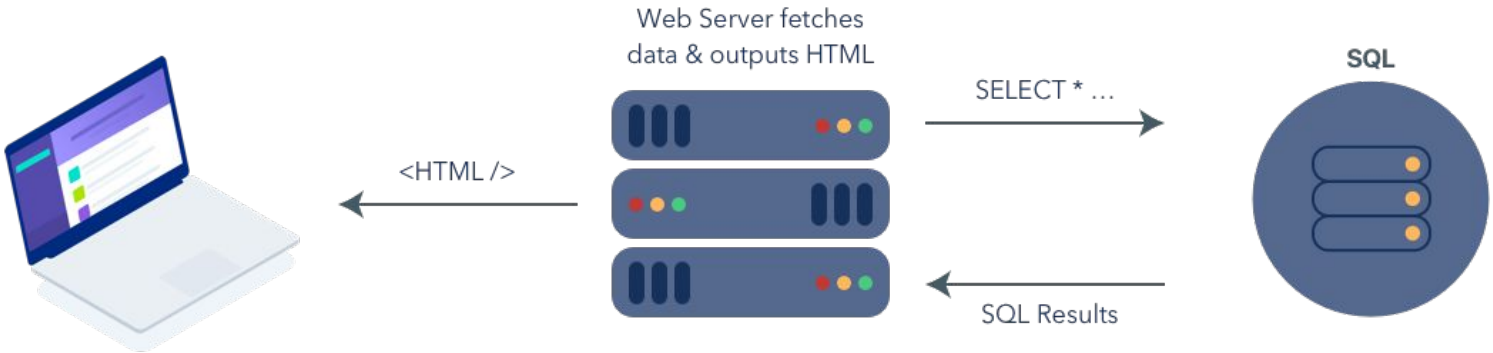
Что: Имплементация подхода CSS-in-JS

Плюсы: Удобный API, производительность

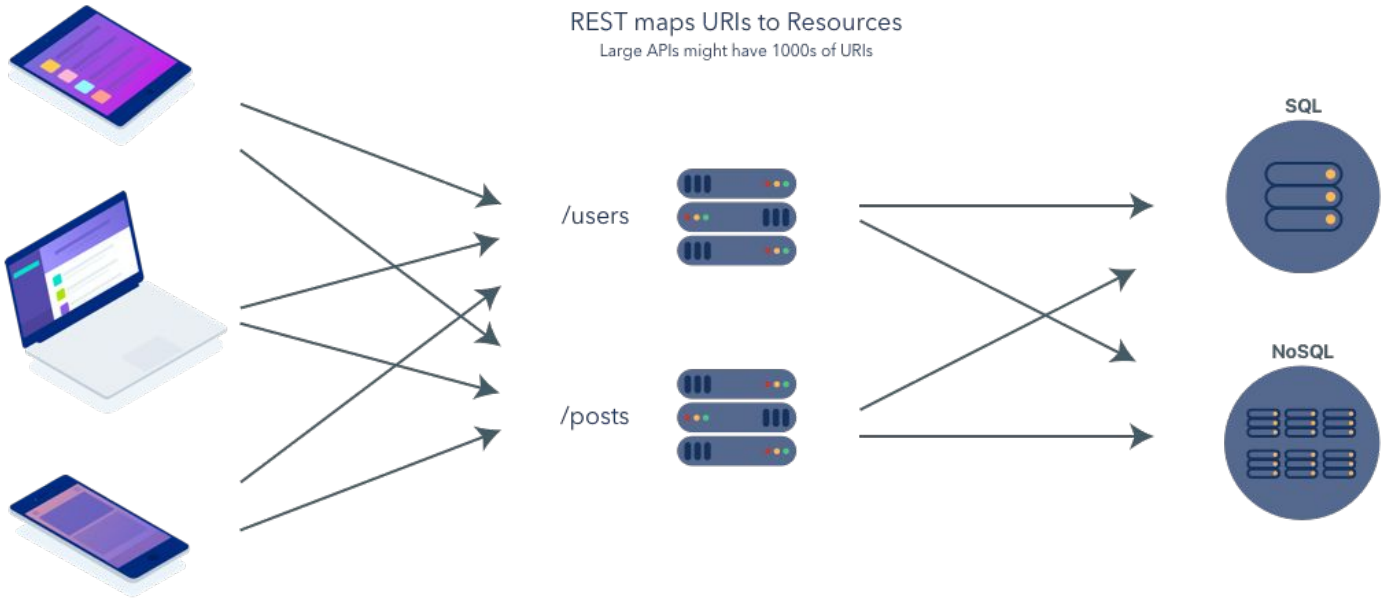
Минусы: Накладные расходы в рантайме

Аналоги: Styled Components, Emotion, Astroturf

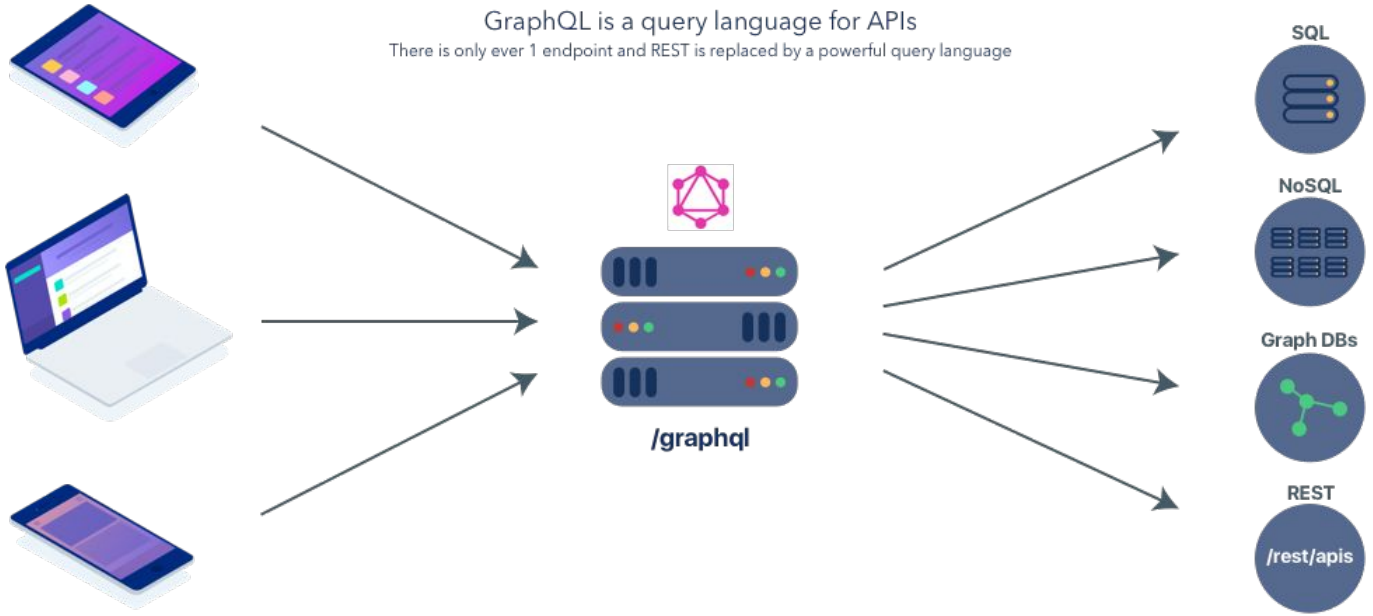
Этап 1. Монолит



Этап 2. REST API

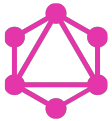


Этап 3. GraphQL API



3. GRAPHQL

Что



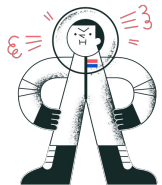
- Язык запросов к API. Не к базам данных. Может работать вообще без них
- Не накладывает ограничений на БД, протокол, языки, библиотеки
- Клиент - запросы, бэкенд - набор функций

Плюсы

- Столько данных, сколько нужно
- Валидация на клиенте и на сервере
- Все связи описаны в схеме
- Самодокументируемость
- Независимость клиента и сервера
- Возможность сложной агрегации на клиенте
- Интроспекция

Минусы

- Не может полностью заменить REST
- Не предназначен для работы с бинарными данными
- Добавляет новые проблемы
- Меняет парадигму и архитектуру
- Невозможность постепенного перехода
- Недостаточная спецификация для обработки ошибок



Аналоги

- JSON:API
- gRPC
- OData
- Logux

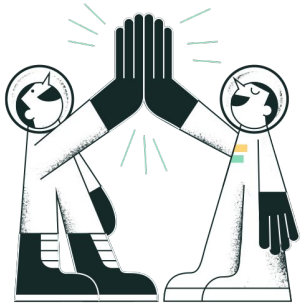


Схема - ядро API

- Query
  - ЧТЕНИЕ

```
query {
  posts(where: {
    title: {
      startsWith: "Что"
    }
  }, orderBy: {
    createdAt: desc
  }) {
    id
    title
    text
  }
}
```

- Mutation
  - ЗАПИСЬ

```
mutation {
  createOnePost(data: {
    title: "Заголовок"
  }) {
    title
    author {
      id
      name
    }
  }
}
```

- Subscription
  - СОБЫТИЯ

```
subscription {
  onCommentCreated(where: {
    post: {
      id: 1
    }
  }) {
    id
    text
    createdAt
  }
}
```

Схема - ядро API

Типы - описание данных и связей

- **Скаляры**
  - Int
  - Float
  - String
  - Boolean
  - ID
  - Кастомный скаляр
- **Interfaces**
- **Unions**
- **Enumerations**

```
type Post {
  id: Int!
  title: String!
  text: String
  author: User
}
```

```
enum PostOrderBy {
  ID_ASC
  ID_DESC
  TITLE_ASC
  TITLE_DESC
}
```

```
type Query {
  post(id: Int!): Post
  posts(
    first: Int = 10
    skip: Int = 0
    orderBy: PostOrderBy = ID_ASC
  ): [Post!]!
}
```

- **GraphQLObjectType**
  - name
  - description
  - fields
    - type
    - description
    - deprecationReason
    - args
    - resolve
- **InputType**

Схема - ядро API

Резолверы - получение данных

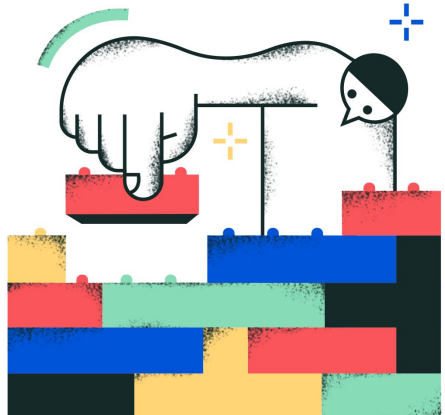
- root (source, parent)
- args
- context
- info

```
Query: {
  post: (root, args, ctx) =>
    ctx.db.post.findOne(where: { id: args.id }),
  posts: (root, { first, skip, orderBy }, { db }) =>
    db.post.findMany(first, skip, orderBy),
  }
```

```
Post: {
  id: (root) => root.id,
  title: (root) => root.title,
  text: (root) => root.text,
  author: ({ authorId }, args, { db }) =>
    db.user.findOne(where: { id: authorId }),
  }
```

4. Создание сервера

1. Prisma
2. Схема данных
3. Базы данных. PostgreSQL
4. Docker и docker-compose
5. Миграции
6. Генерация клиента Prisma
7. Построение GraphQL-схемы
8. Nexus Schema
9. Graphql-сервер
10. GraphQL Yoga



CSSSR

Prisma

Что: Фреймворк для работы с БД. Из коробки: генерация БД по схеме, миграции, ORM для доступа к данным, панель администрирования

Плюсы: Типизация, активное развитие, документация

Минусы: Активное развитие

Аналоги: Postgraphile, Hasura

Nexus Schema

Что: Конструктор GraphQL-схем

Плюсы: Типизация, интеграция с Prisma, выразительный и декларативный

Минусы: Порог входа, поддержка Windows

Аналоги: GraphQL-compose, type-GraphQL

4. СОЗДАНИЕ СЕРВЕРА

GraphQL Yoga

Что: GraphQL-сервер

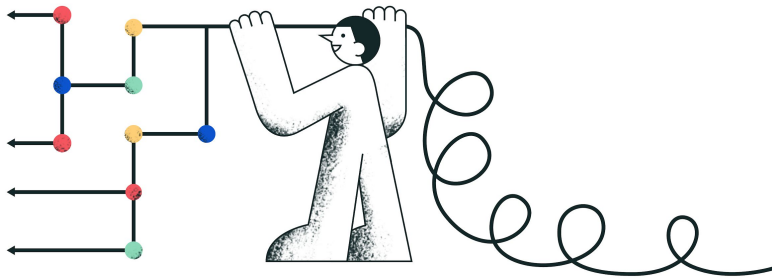
Плюсы: Много возможностей

Минусы: Не обновляется

Аналоги: Apollo Server

5. Получение данных

1. Apollo Client 3
2. Получение, обработка и хранение ответов
3. Подключение к Next.js
4. Особенности работы с SSR
5. Написание запросов



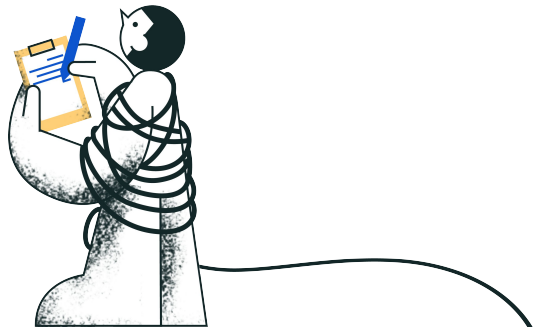
GraphQL-клиенты

Запросы:

- Протокол
- Обработка ошибок
- Дедупликация
- Повторение
- Объединение

Кеш:

- Нормализация
- Сборка мусора
- Подписка на обновления
- Ручное изменение



Apollo Client

Что: GraphQL-клиент

Плюсы: Расширяемость с помощью плагинов, мощный кеш, большое и отзывчивое сообщество

Минусы: Много багов

Аналоги: Relay, Urql, GQLess

6. Улучшаем клиент

1. Интроспекция
2. Кодогенерация
3. GraphQL Codegen
4. Фрагменты

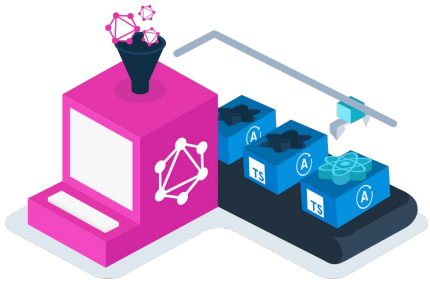
GraphQL Codegen

Что: Генератор типов, запросов и мутаций

Плюсы: Широкая функциональность, расширяемость плагинами, поддержка последних версий Apollo

Минусы: Не выявлены

Аналоги: Relay Compiler, Apollo CLI



7. Прокачиваем сервер

1. Мутации
2. Ручное обновление кэша
3. Авторизация, аутентификация
4. JWT
5. Передача токена через куки
6. Резолверы для авторизации
7. Контекст запроса
8. Computed inputs

CSSSR

Похожие проекты

1. Blitz.js
2. RedwoodJS