Talantix

GraphQL для взрослых: как не влипнуть в молодежные фреймворки и внедрить технологию в старый проект

Илья Горский

Ведущий разработчик hh.ru

Илья Горский

Ведущий разработчик hh.ru

- 8 лет в разработке, последние 4 года в hh.ru

Илья Горский

Ведущий разработчик hh.ru

- 8 лет в разработке, последние 4 года в hh.ru
- Работал в СМИ, стартапах и агентствах

Илья Горский

Ведущий разработчик hh.ru

- 8 лет в разработке, последние 4 года в hh.ru
- Работал в СМИ, стартапах и агентствах
- Ежегодно читаю лекции по TypeScript для студентов «Школы программистов hh.ru»

Илья Горский

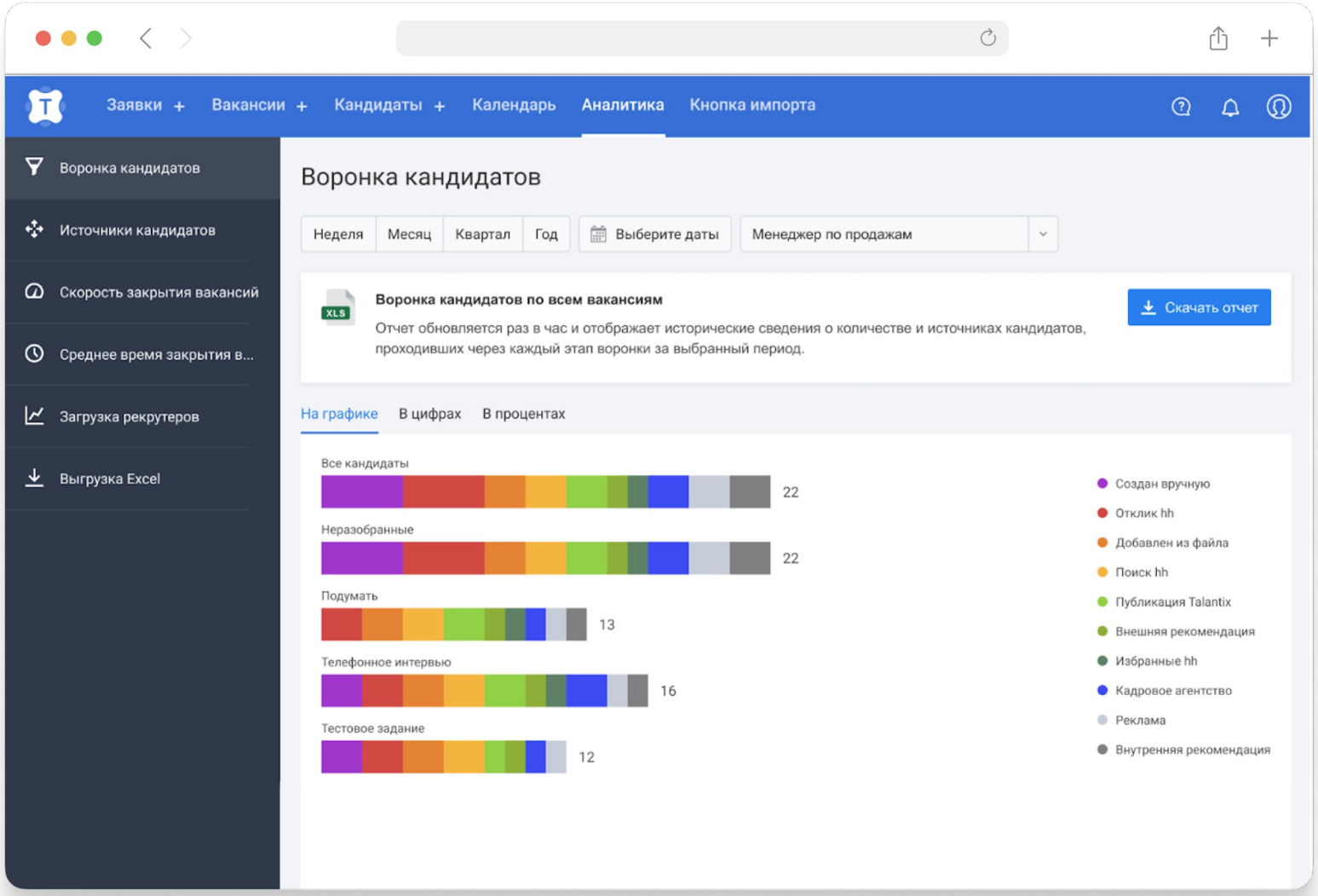
Ведущий разработчик hh.ru

- 8 лет в разработке, последние 4 года в hh.ru
- Работал в СМИ, стартапах и агентствах
- Ежегодно читаю лекции по TypeScript для студентов «Школы программистов hh.ru»
- Увлекаюсь архитектурой и проектированием дизайн-систем

О продукте

Talantix

Облачная CRM-система
для ведения процесса подбора
для клиентов среднего-крупного
сегмента



Чего не будет в докладе?

Чего не будет в докладе?

1

Детального сравнения фреймворков

Чего не будет в докладе?

1

Детального сравнения фреймворков

2

Обработки ошибок

Чего не будет в докладе?

1

Детального сравнения фреймворков

2

Обработки ошибок

3

Рассказ про Backend-for-Frontend и бэкенд-сервер

Чего не будет в докладе?

1

Детального сравнения фреймворков

2

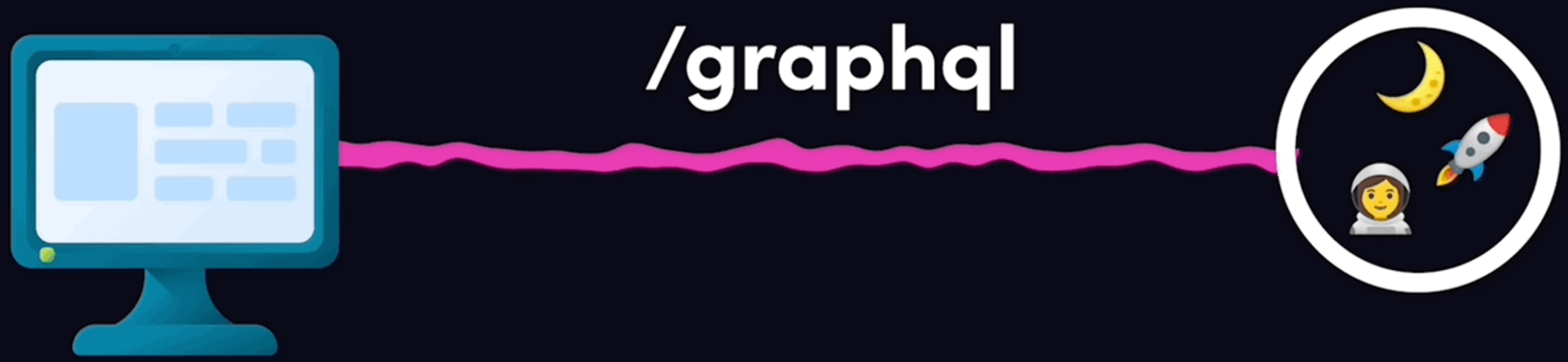
Обработки ошибок

Что такое GraphQL?

Это Gateway

Rest

GraphQL Gateway



underFETCHING

OVERfetching

Query

schema.gql schema.gql

```
type Video {
    url: String!
    creator: Creator
}
```

QUERY

entry point to
read data

Mutation

schema.gql schema.gql

```
type Query {
    videos: [Video]
    creator(id: String!): Creator
}
```

Request

CLIENT

REQUEST

query.gql query.gql

```
query getRockets {
    Execute Query
    rocket {
        name
        thrust
        captain {
            name
            callsign
        }
    }
}
```

SERVER

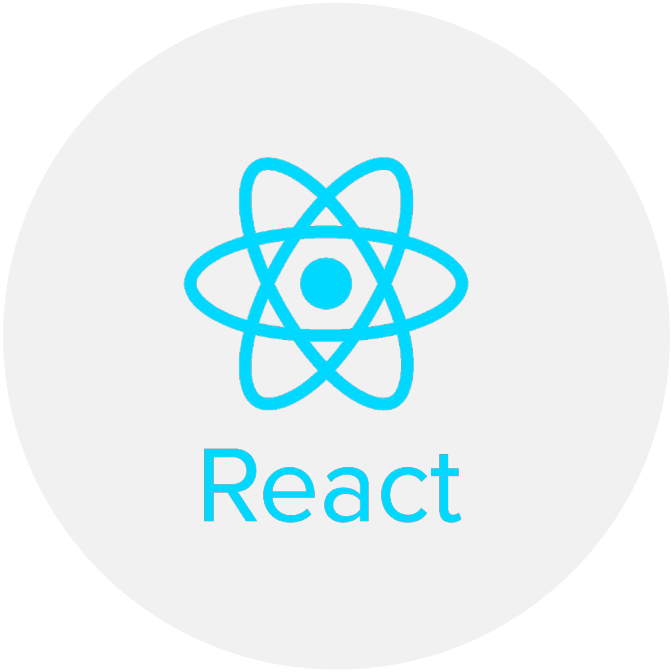
RESOLVE

{ } response.json response.json\...

```
{
    "name": "Saturn V",
    "thrust": 7891000,
    "captain": {
        "name": "Neil Armstrong",
        "callsign": "Neil"
    }
}
```

О проекте

Frontend



Backend

Как работаем с асинхронными запросами

Асинхронные экшены

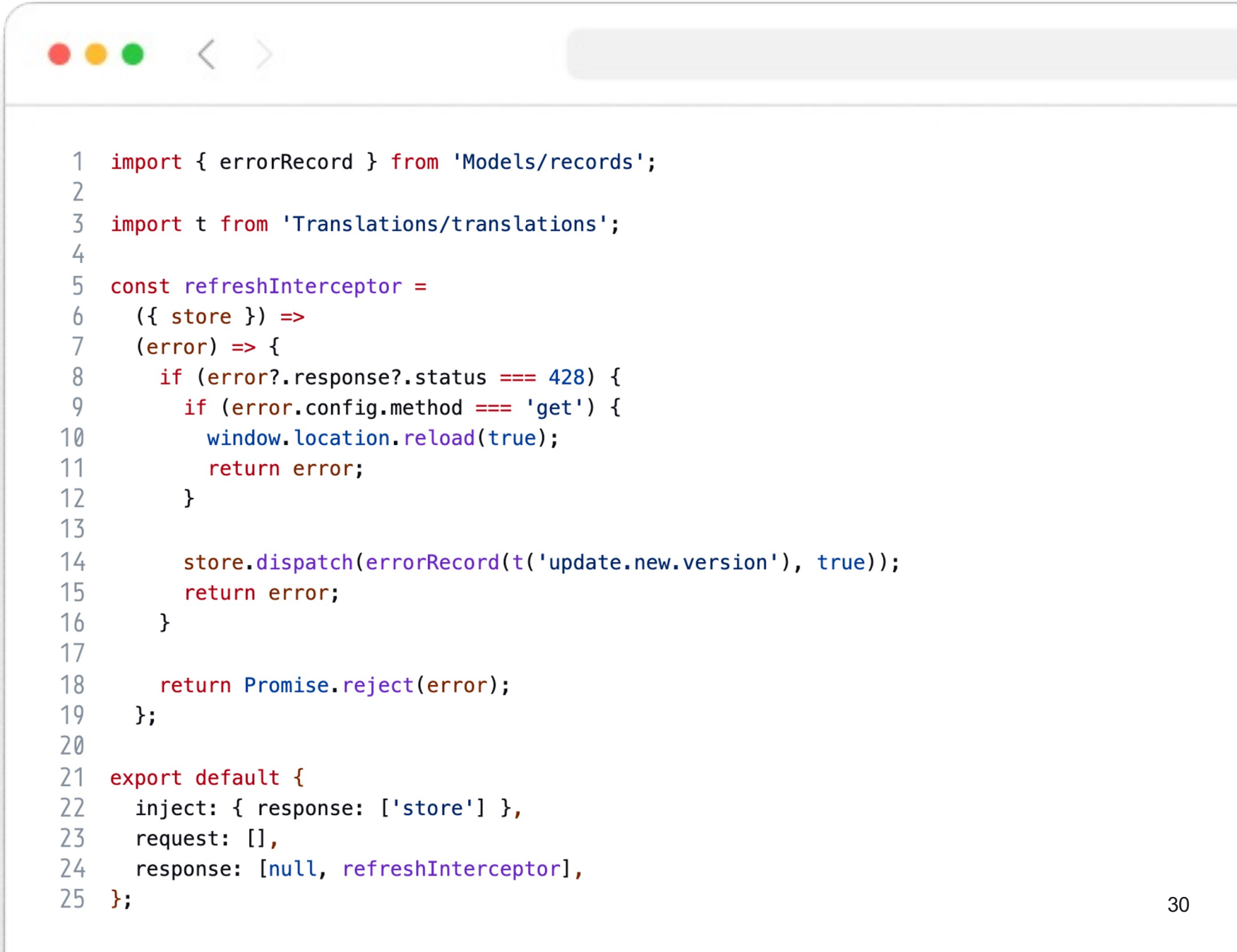
personsMap.js

Users > i.gorskiy > projects > tms > tms-static > app > actions > ats > personsMap > personsMap.js

```
108  addReaction(
172  );
173
174  addReaction(
175    PERSONS_MAP_LOAD,
176    async (
177      { getState, dispatch, fetcher },
178      { vacancyId, statusId, offset, query, resetPersonData, mergePersonId, fromVacancy }
179    ) => {
180      const isMergeRequest = mergePersonId !== null;
181
182 >    if ((offset === 0 || isMergeRequest) && resetPersonData !== false) {…
184      }
185
186 >    if (resetPersonData) {…
188      }
189
190      dispatch(fetchPersons());
191
192      try {
193 >      const { data } = await fetcher.get(getUrl({ vacancyId, statusId, mergePersonId }), {     prizemlenie, 6 years ago
196        });
197
198        const actions = [];
199
200 >      if (offset === 0 && resetPersonData !== false) {…
204        }
205
```

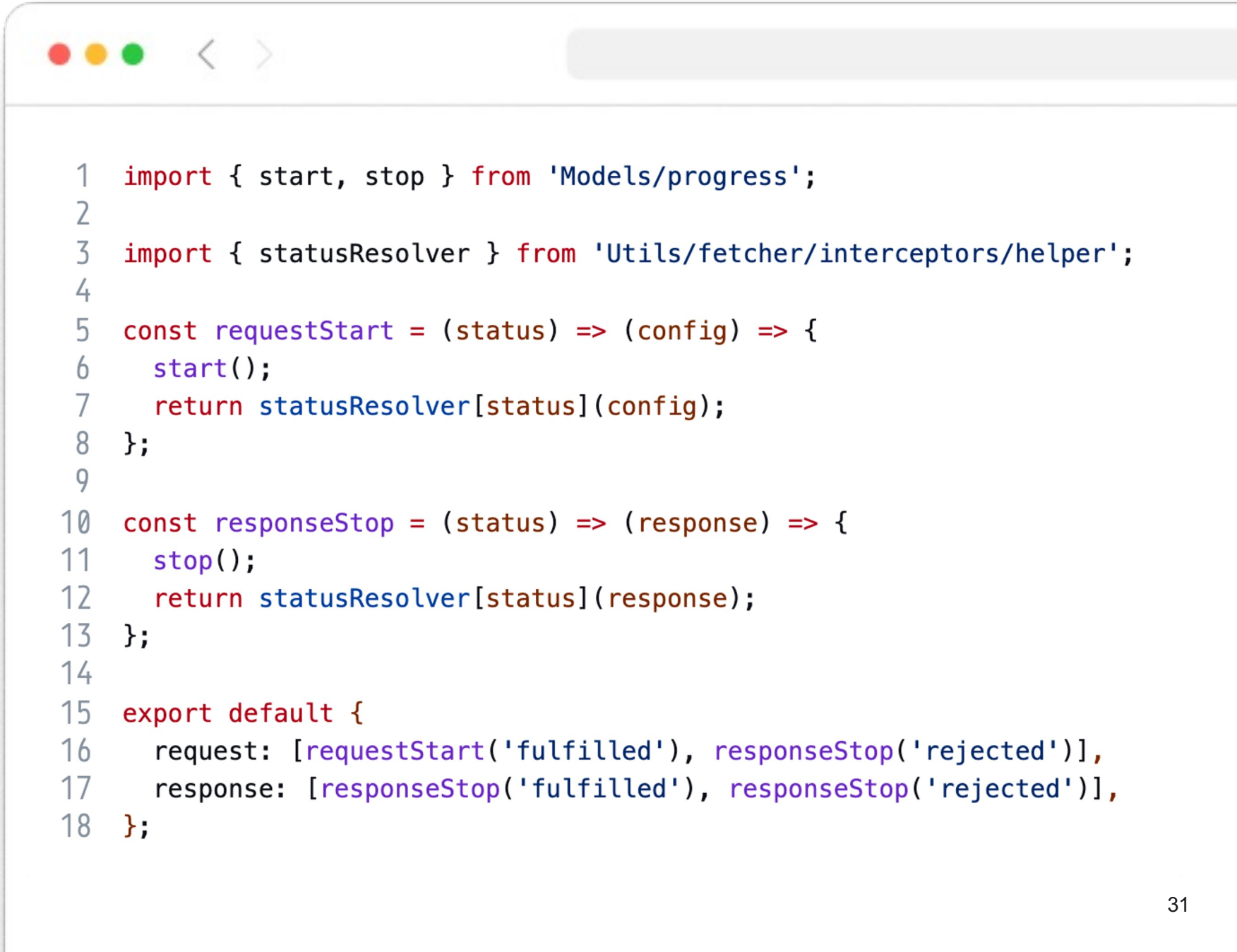
Используем Interceptors

Interceptors



```
import { errorRecord } from 'Models/records';

import t from 'Translations/translations';

const refreshInterceptor =
  ({ store }) =>
  (error) => {
    if (error?.response?.status === 428) {
      if (error.config.method === 'get') {
        window.location.reload(true);
        return error;
      }

      store.dispatch(errorRecord(t('update.new.version'), true));
      return error;
    }

    return Promise.reject(error);
  };

export default {
  inject: { response: ['store'] },
  request: [],
  response: [null, refreshInterceptor],
};
```

Interceptors



```
import { start, stop } from 'Models/progress';

import { statusResolver } from 'Utils/fetcher/interceptors/helper';

const requestStart = (status) => (config) => {
  start();
  return statusResolver[status](config);
};

const responseStop = (status) => (response) => {
  stop();
  return statusResolver[status](response);
};

export default {
  request: [requestStart('fulfilled'), responseStop('rejected')],
  response: [responseStop('fulfilled'), responseStop('rejected')],
};
```

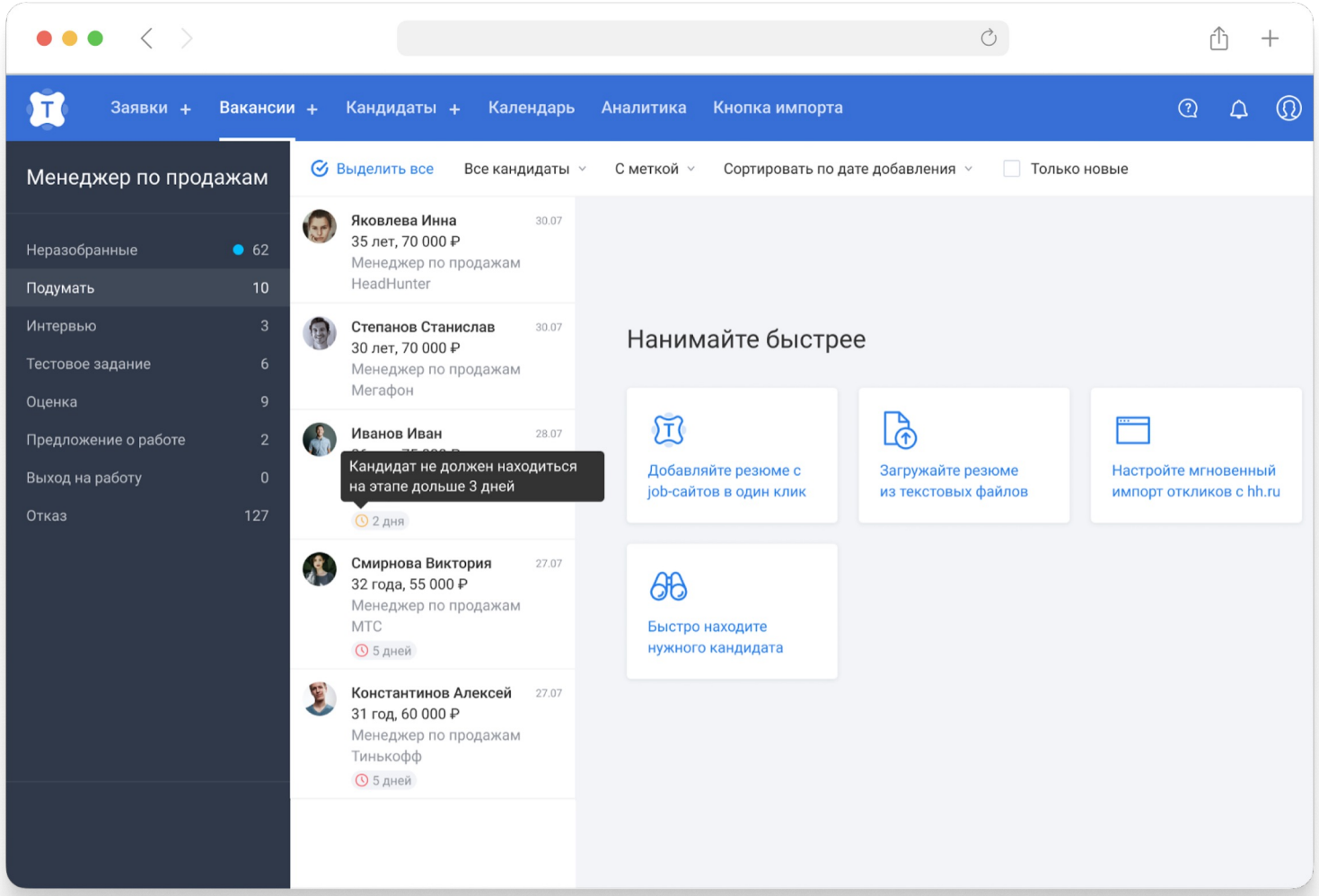
Постановка проблемы

Проблемы текущего подхода

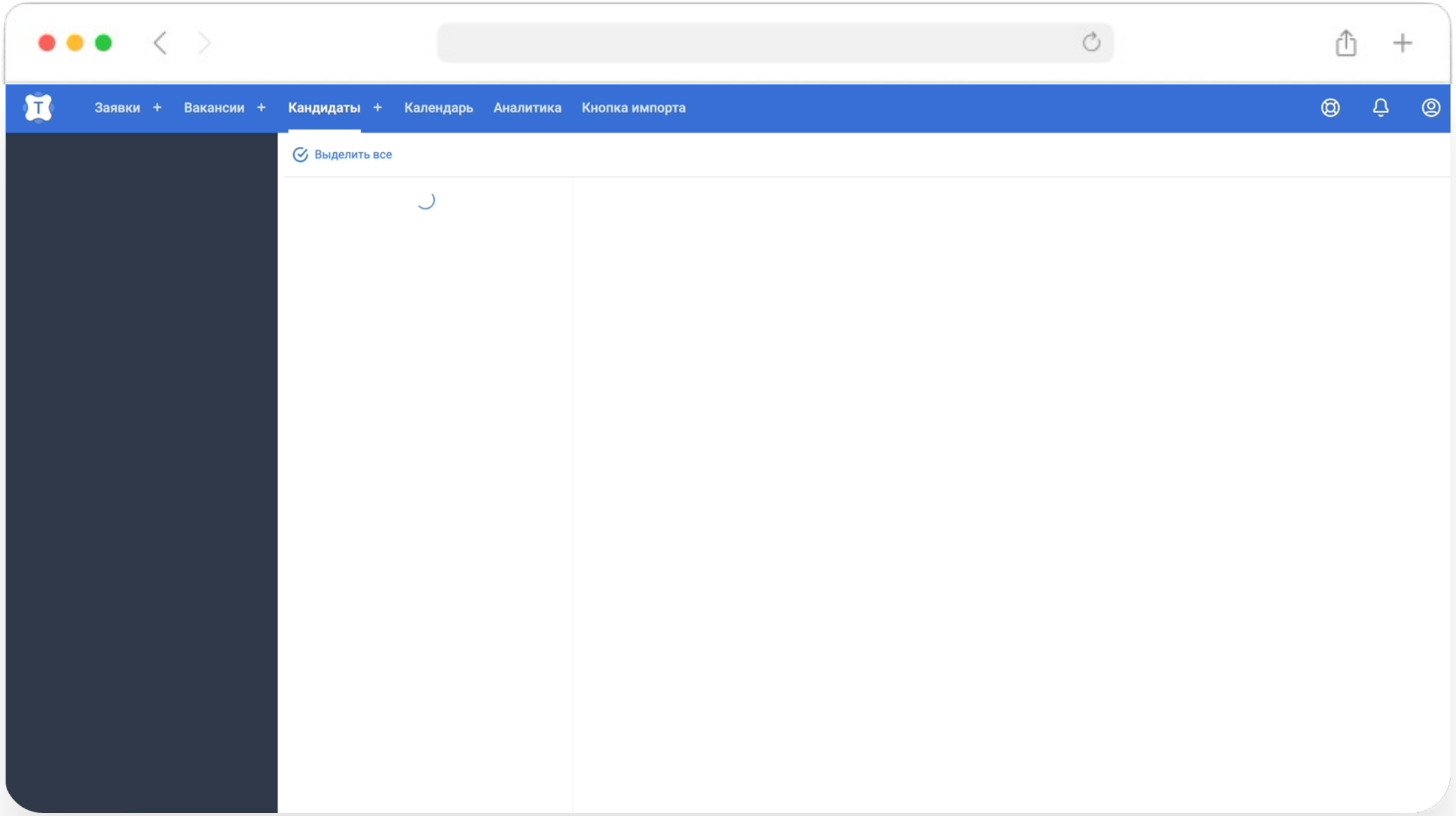
1

AJAX страницы

Одним запросом получаем все данные для страницы



Список кандидатов

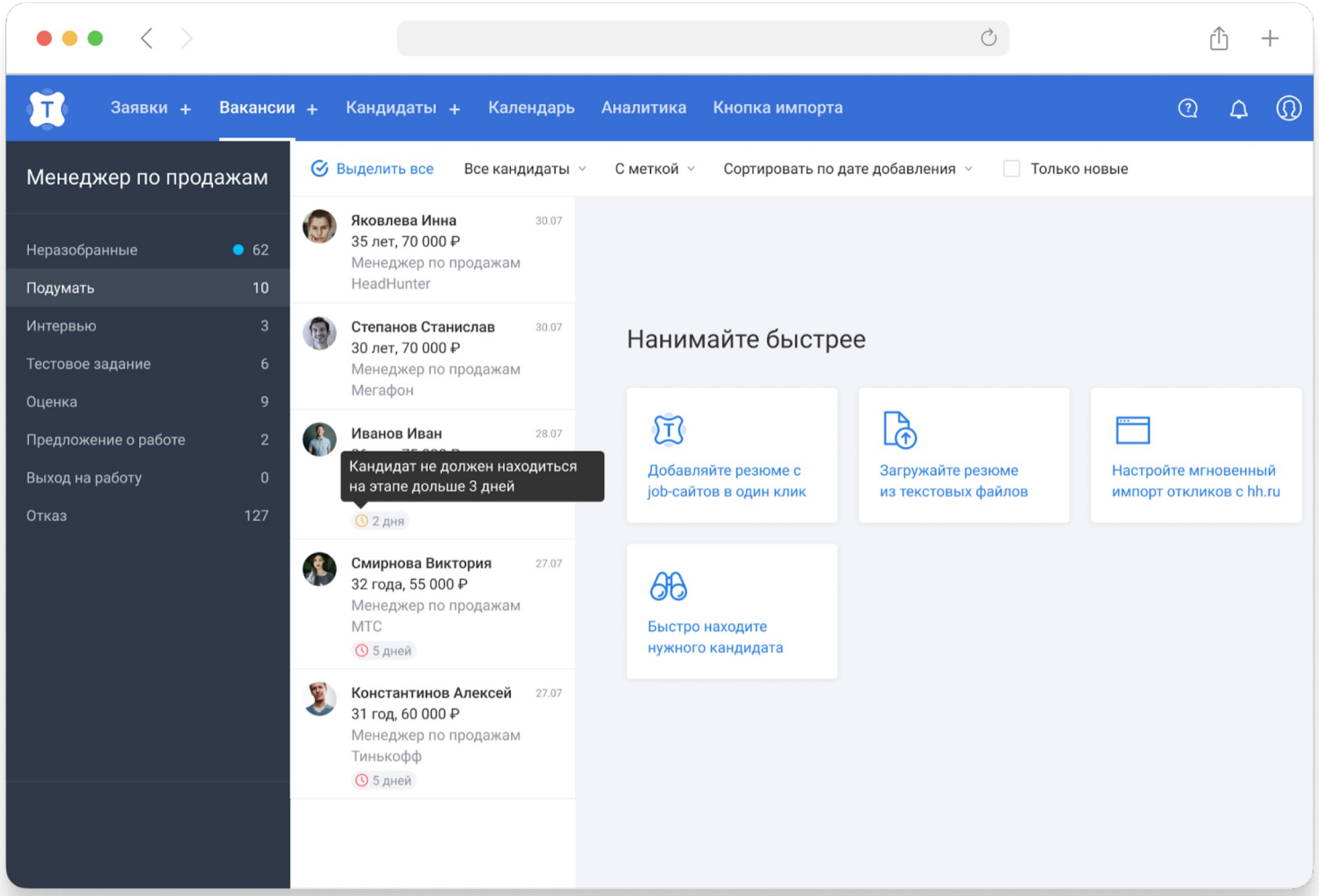


Проблемы текущего подхода

1. **AJAX страницы**

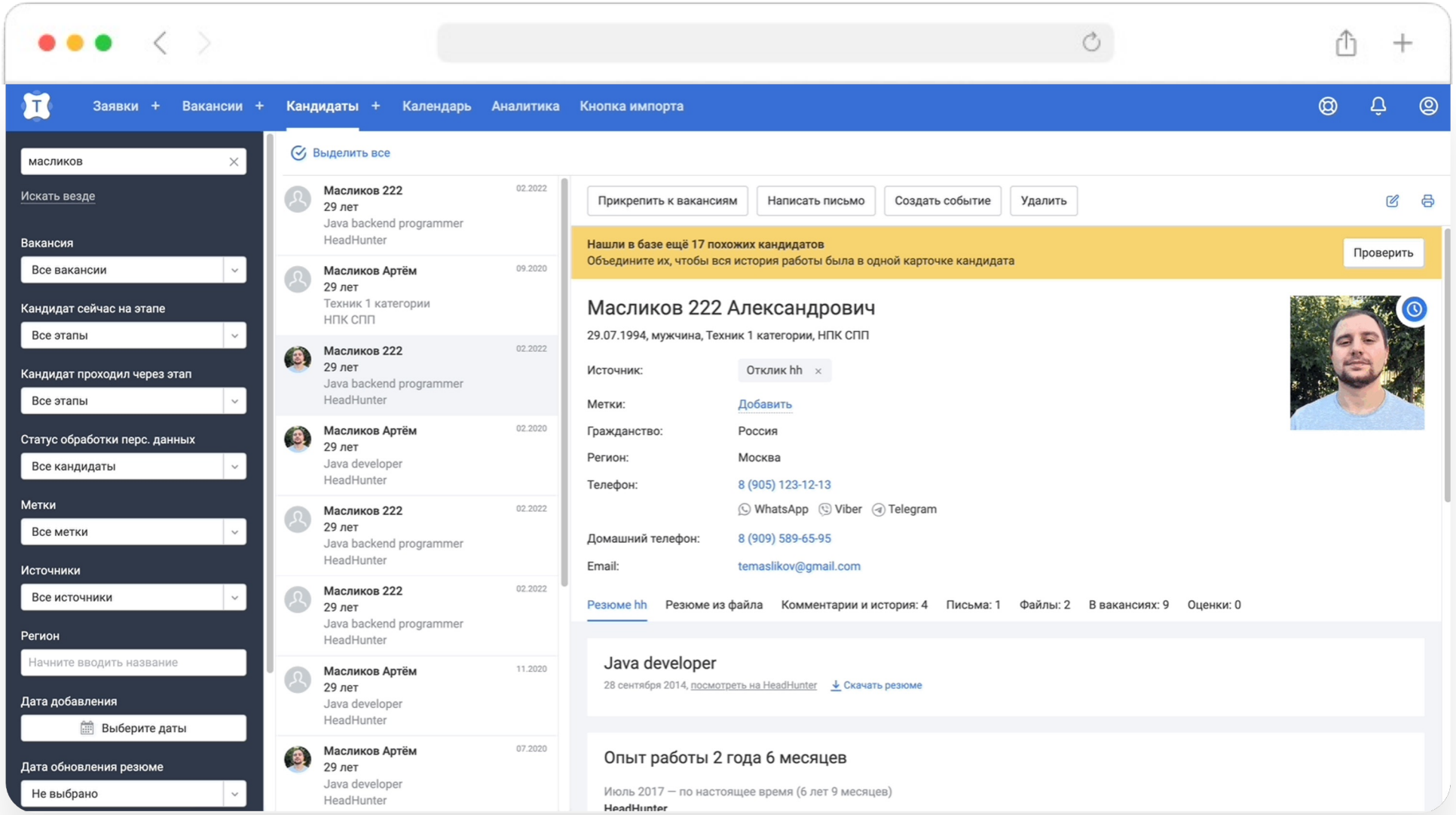
   Одним запросом получаем все данные для страницы

2. **Переизбыток данных**



Пример трех полей

Запрос на страницу кандидатов ради трех полей



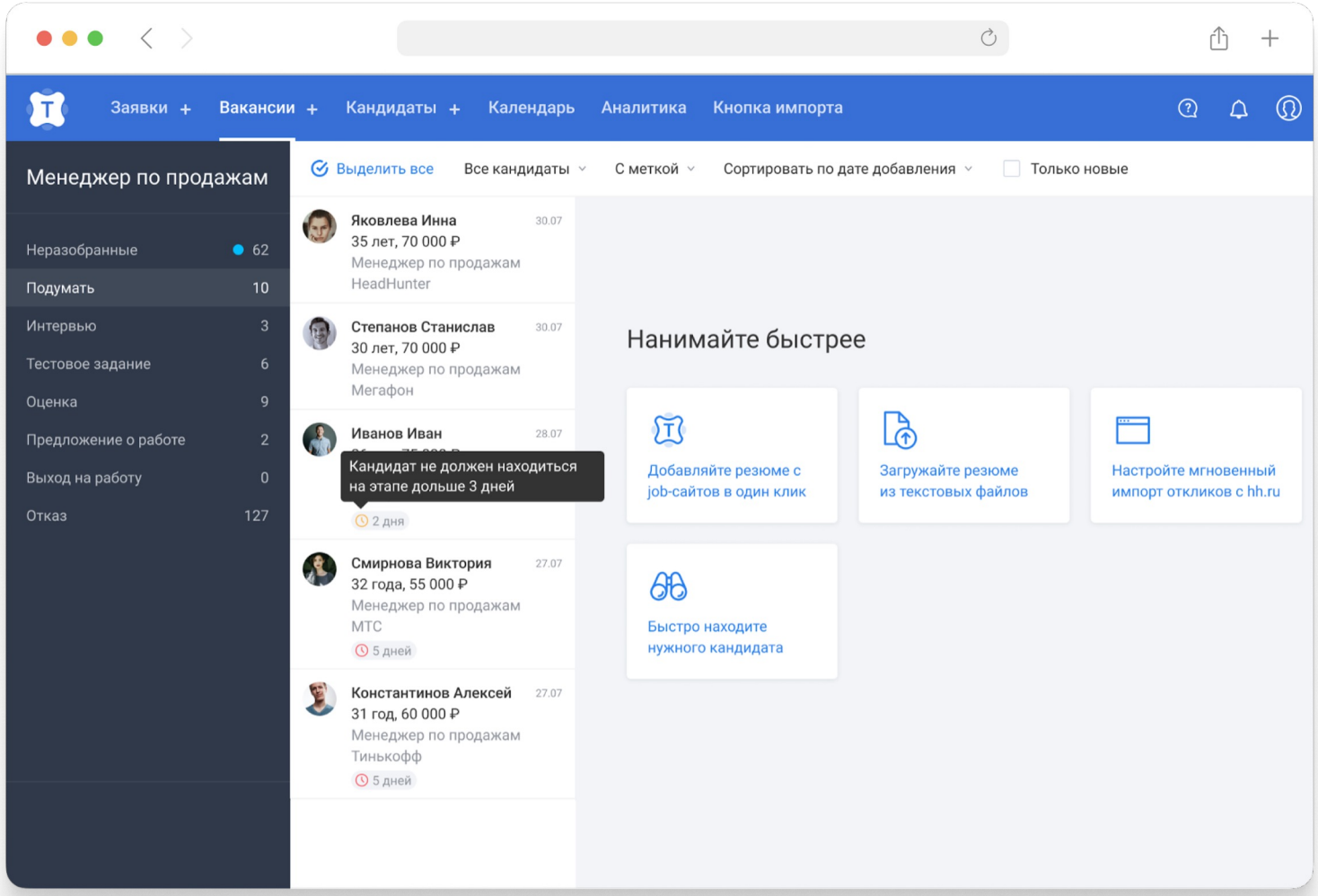
Проблемы текущего подхода

1. **AJAX страницы**

   Одним запросом получаем все данные для страницы

2. **Переизбыток данных**

   Мы столкнулись с **Overfetching**. Попытка рефакторинга заденет другие части приложения/



Проблемы текущего подхода

1. **AJAX страницы**

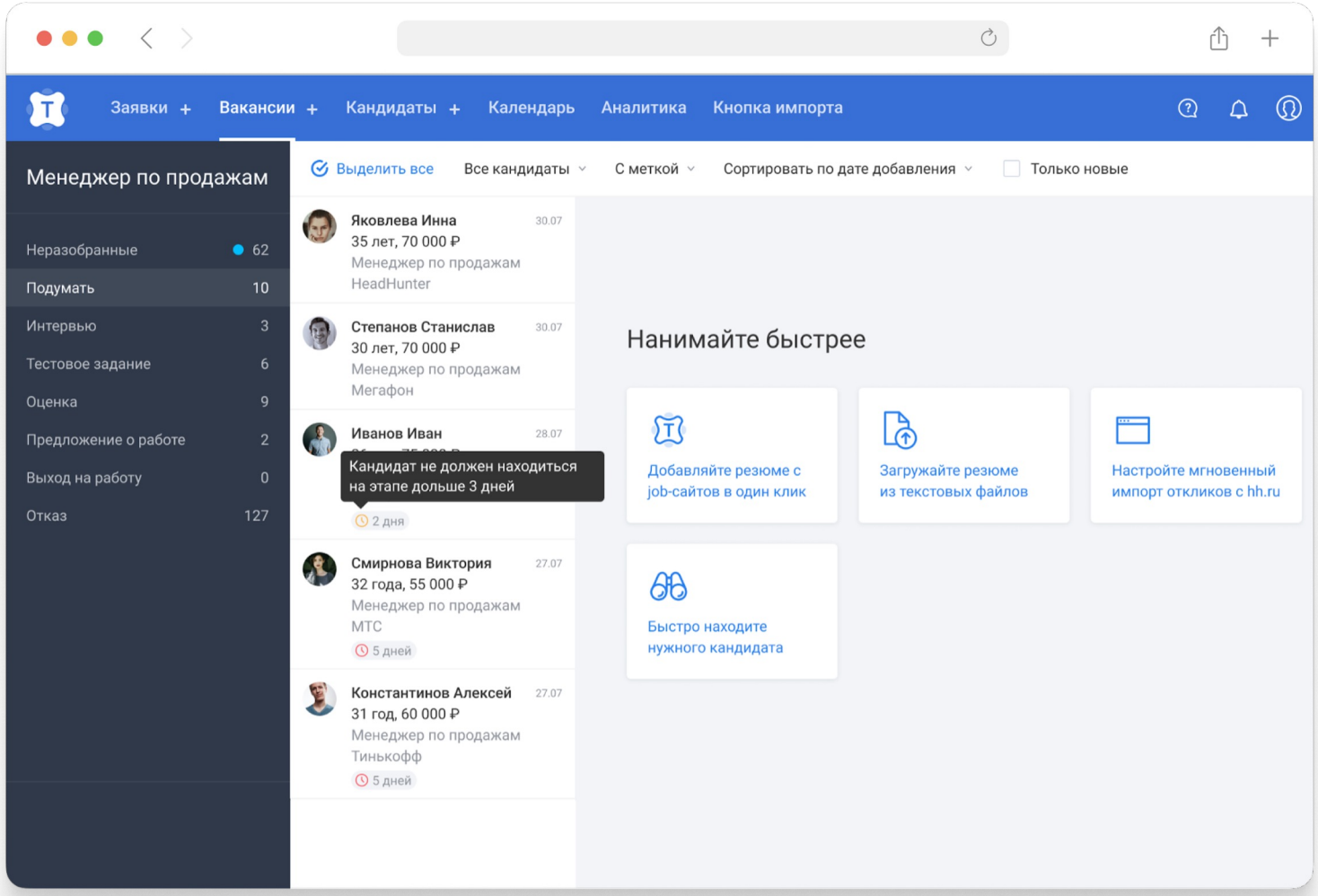
   Одним запросом получаем все данные для страницы

2. **Переизбыток данных**

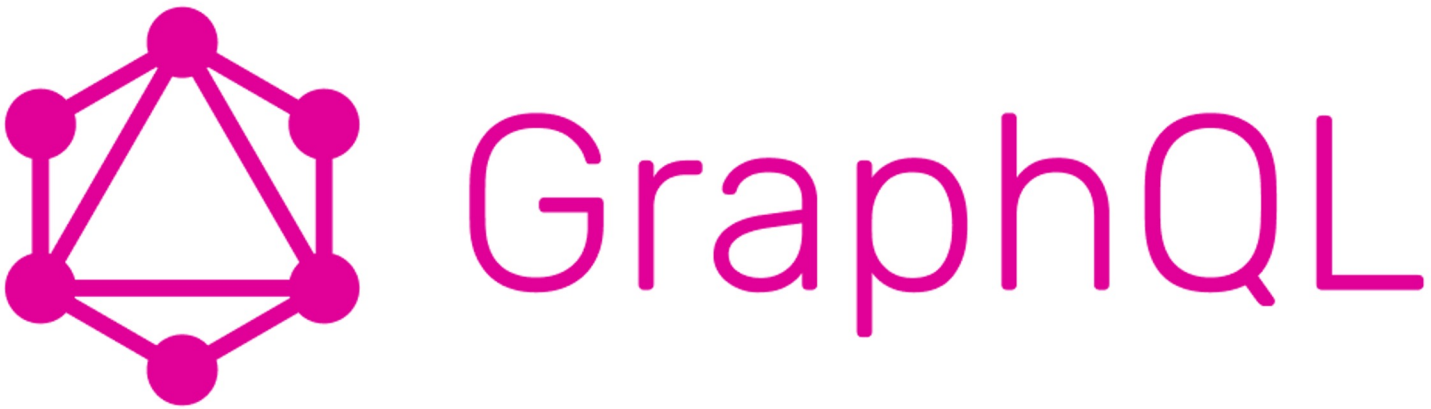
   Мы столкнулись с **Overfetching**. Попытка рефакторинга заденет другие части приложения

3. **Водопад данных**

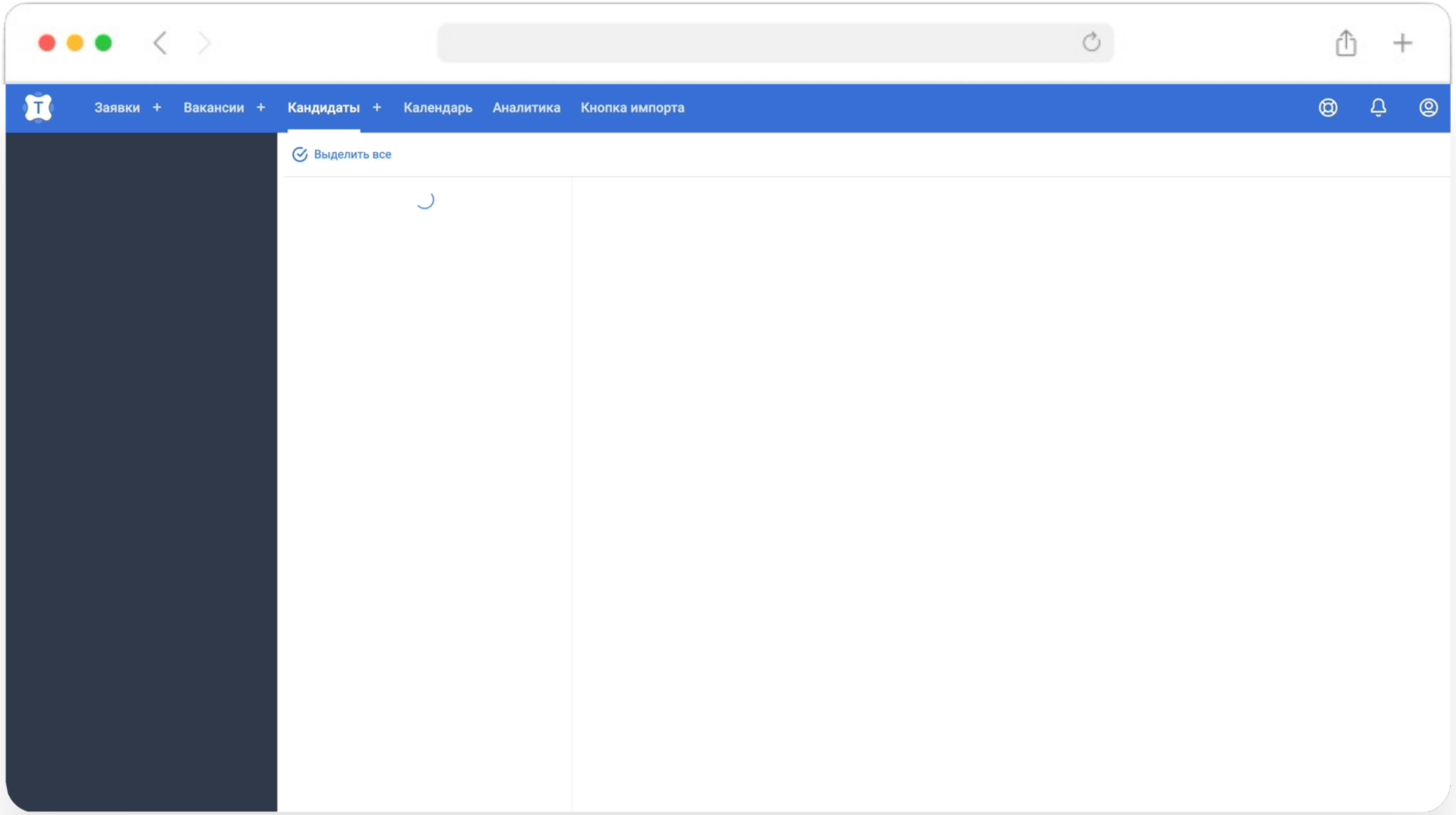
   Делаем + N запросов за сущностями или добавляем еще данных в **AJAX** ручку



Похоже нам нужен...


GraphQL

Пробуем на странице кандидатов



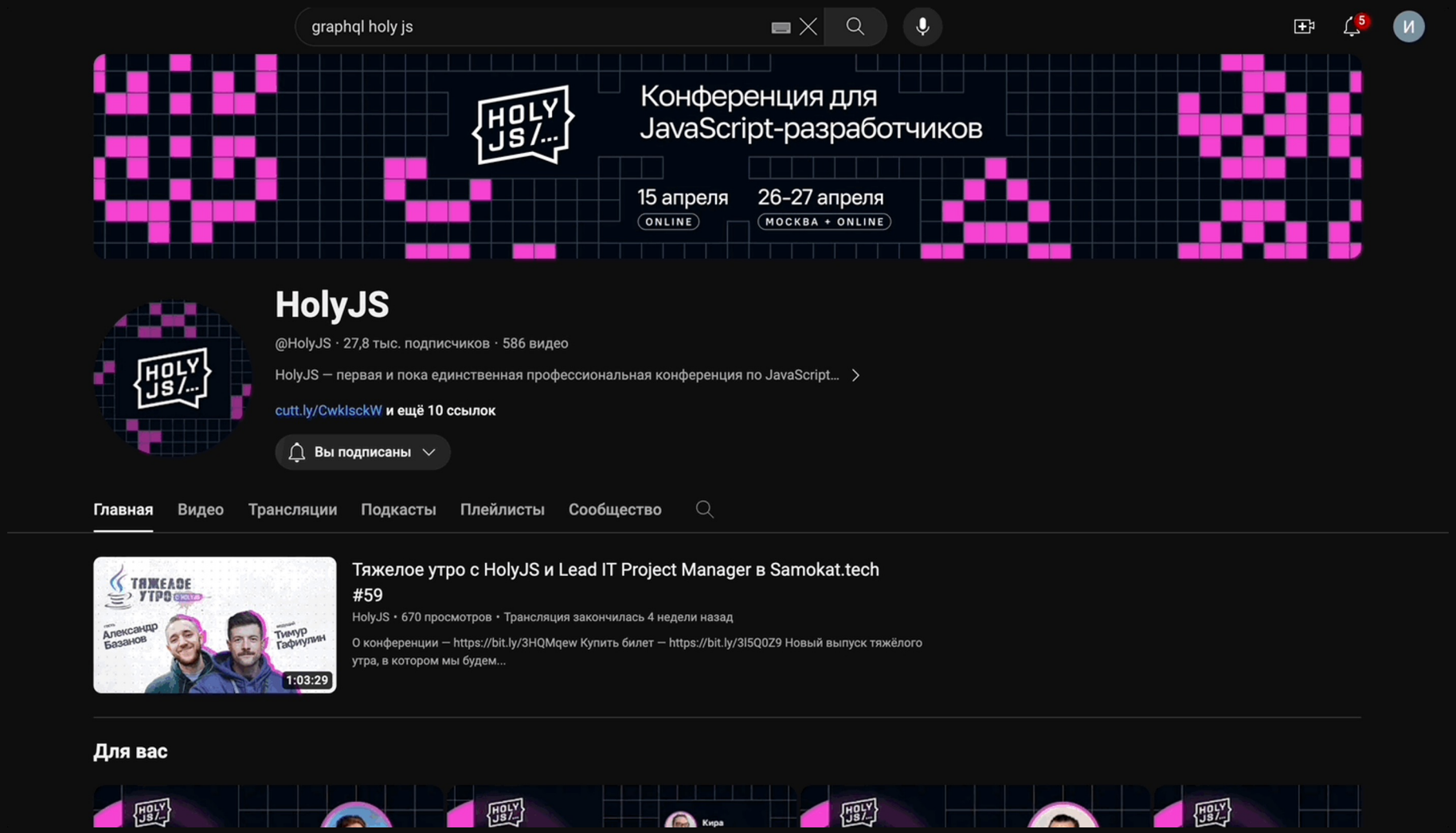
Ресерч или приключение на 20 минут


Let's go. In and out. Twenty minute adventure.

С чего начинается ресерч?

С чего начинается ресерч?

С докладов на HolyJS!



Фреймворк

Relay

18.2k

Фреймворк

Relay

18.2k

+

Преимущества

- Мощный инструмент (компилятор, фрагменты, стор)

Фреймворк

Relay

18.2k

+

Преимущества

- Мощный инструмент (компилятор, фрагменты, стор)

–

Недостатки

- Сильное влияние на архитектуру бэкенда и фронтенда, сложно начать работу
- Маленькое комьюнити
- Подхождит только для React

Фреймворк

Relay

18.2k

+

Преимущества

- Мощный инструмент (компилятор, фрагменты, стор)

–

Недостатки

- Сильное влияние на архитектуру бэкенда и фронтенда, сложно начать работу
- Маленькое комьюнити
- Подходжит только для React

Фреймворк

Apollo

☆ 19.2k

Фреймворк

Apollo

19.2k

+

Преимущества

- Большое комьюнити
- Легкий старт
- Целая экосистема (Apollo client, Apollo studio, Apollo server)
- Хорошая документация
- Биндинги под все фреймворки

Фреймворк

Apollo

19.2k

+

Преимущества

- Большое комьюнити
- Легкий старт
- Целая экосистема (Apollo client, Apollo studio, Apollo server)
- Хорошая документация
- Биндинги под все фреймворки

–

Недостатки

- Энтерпрайзность

Фреймворк

Apollo

19.2k

+

Преимущества

- Большое комьюнити
- Легкий старт
- Целая экосистема (Apollo client, Apollo studio, Apollo server)
- Хорошая документация
- Биндинги под все фреймворки

–

Недостатки

- Энтерпрайзность

Фреймворк

URQL

☆ 8.5k

Библиотека

URQL

☆ 8.5k

+

Преимущества

- Настраиваемая библиотека
- Есть биндинги под любой фреймворк
- Отличная документация
- Альтернативный взгляд на работу с GraphQL

Библиотека

URQL

☆ 8.5k

\+

Преимущества

- Настраиваемая библиотека
- Есть биндинги под любой фреймворк
- Отличная документация
- Альтернативный взгляд на работу с GraphQL

–

Недостатки

- Маленькое, но дружное комьюнити

Библиотека

URQL

☆ 8.5k

\+

Преимущества

- Настраиваемая библиотека
- Есть биндинги под любой фреймворк
- Отличная документация
- Альтернативный взгляд на работу с GraphQL

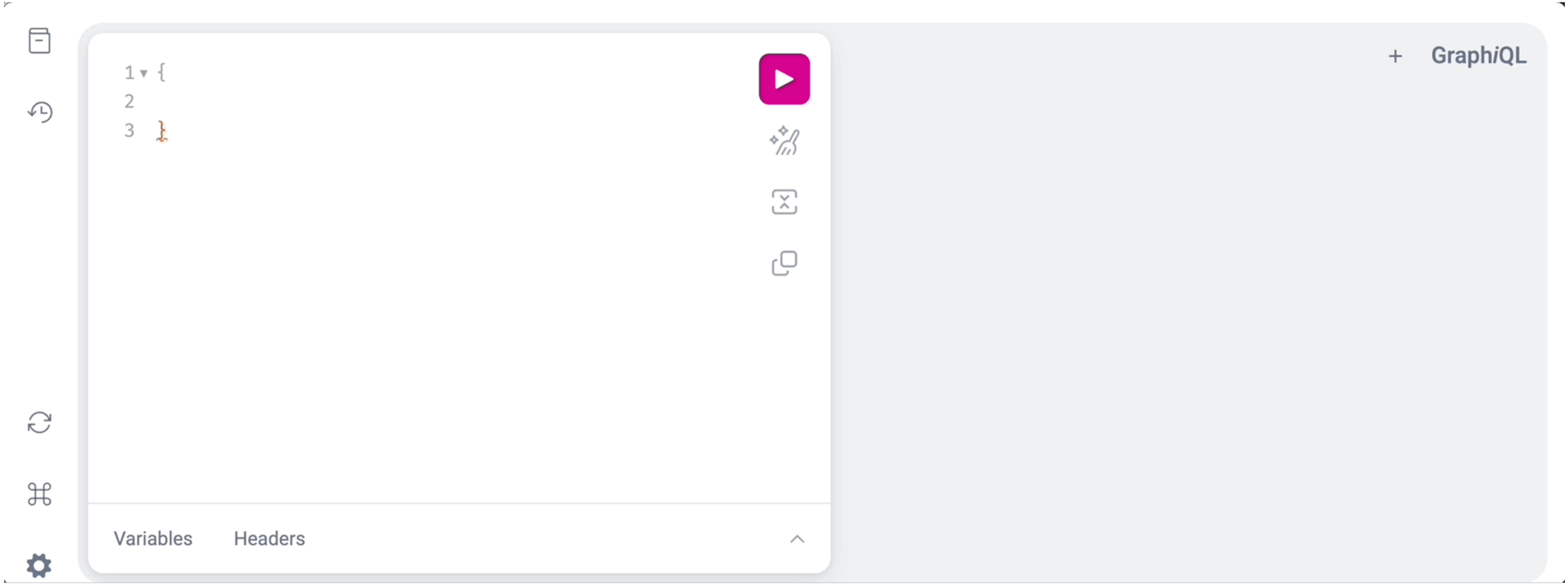
–

Недостатки

- Маленькое, но дружное комьюнити

Фичи

Фича: плейграунд



Фича: типы на схему

types.ts M

src > __generated__ > types.ts > Country

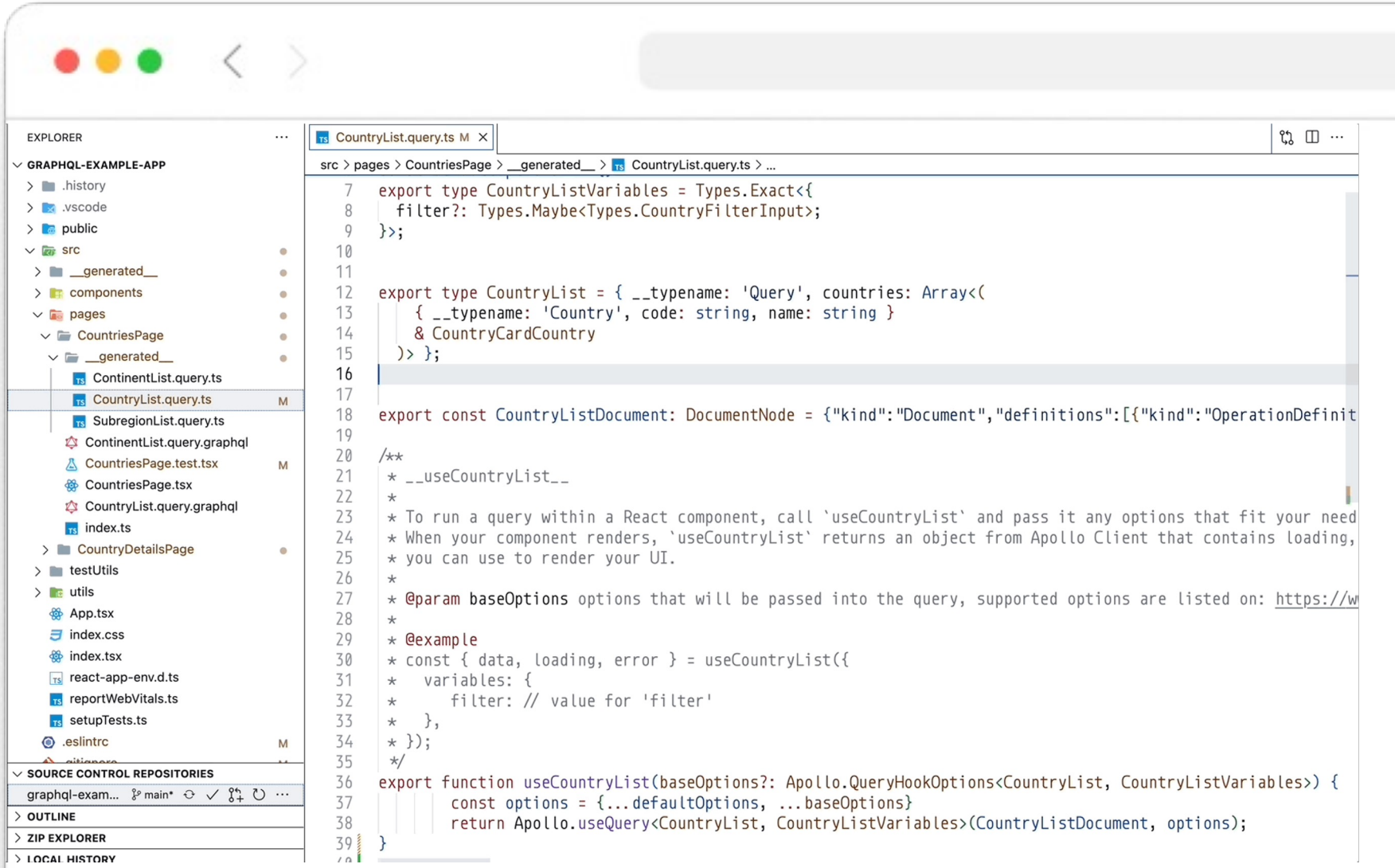
```ts
export type Continent = {
  __typename?: 'Continent';
  code: Scalars['ID'];
  countries: Array<Country>;
  name: Scalars['String'];
};

export type ContinentFilterInput = {
  code?: Maybe<StringQueryOperatorInput>;
};

export type Country = {
  __typename?: 'Country';
  awsRegion: Scalars['String'];
  capital?: Maybe<Scalars['String']>;
  code: Scalars['ID'];
  continent: Continent;
  currencies: Array<Scalars['String']>;
  currency?: Maybe<Scalars['String']>;
  emoji: Scalars['String'];
  emojiU: Scalars['String'];
  languages: Array<Language>;
  name: Scalars['String'];
  native: Scalars['String'];
  phone: Scalars['String'];
  phones: Array<Scalars['String']>;
  states: Array<State>;
  subdivisions: Array<Subdivision>;
};


export type CountrynameArgs = {
  lang?: Maybe<Scalars['String']>;
};
```

schema.graphql M

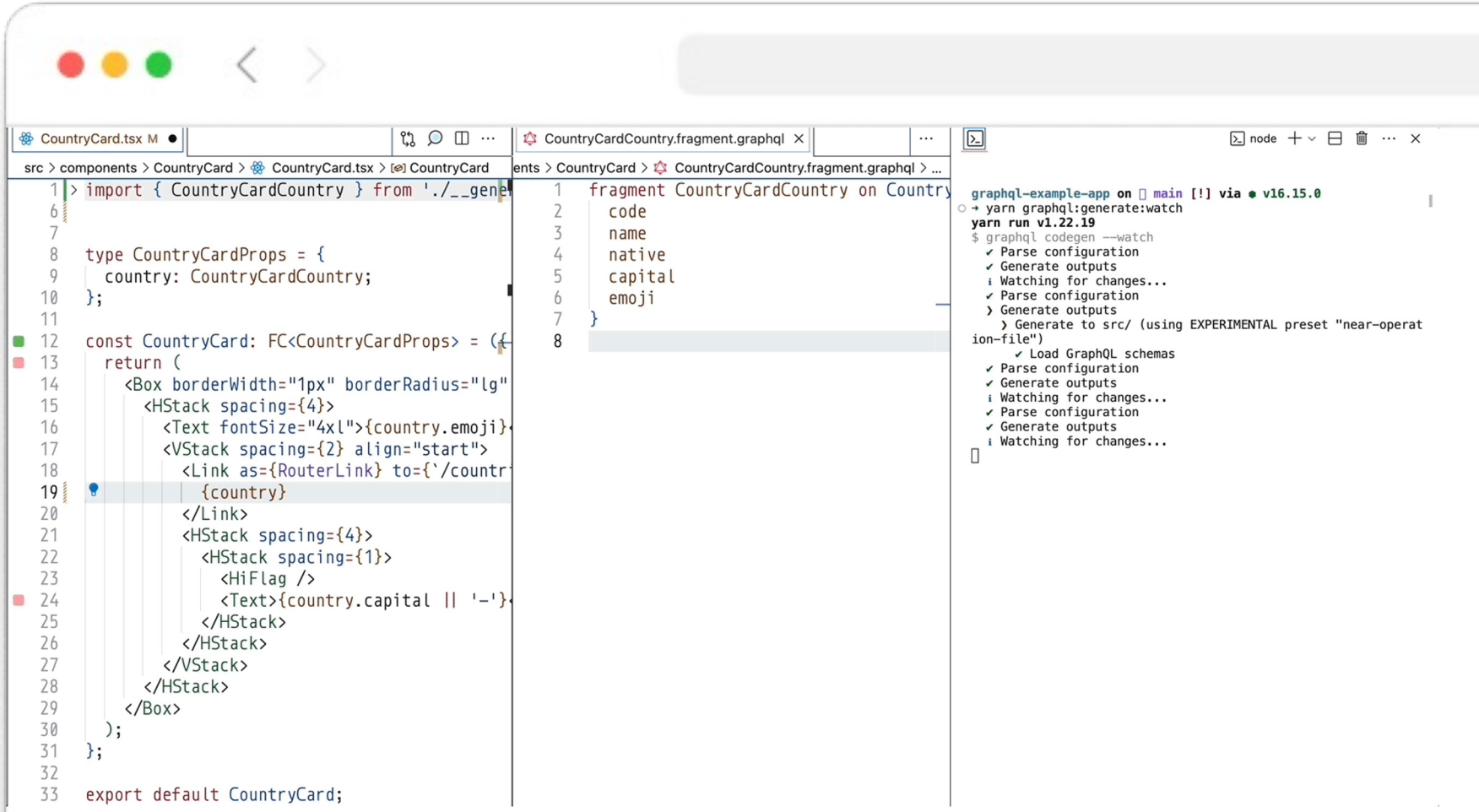
schema.graphql > ...

```graphql
directive @cacheControl(maxAge: Int, scope: CacheControlScope) on FIELD_

type Continent {
  code: ID!
  countries: [Country!]!
  name: String!
}

input ContinentFilterInput {
  code: StringQueryOperatorInput
}

type Country {
  awsRegion: String!
  capital: String
  code: ID!
  continent: Continent!
  currencies: [String!]!
  currency: String
  emoji: String!
  emojiU: String!
  languages: [Language!]!
  name(lang: String): String!
  native: String!
  phone: String!
  phones: [String!]!
  states: [State!]!
  subdivisions: [Subdivision!]!
}

input CountryFilterInput {
  code: StringQueryOperatorInput
  continent: StringQueryOperatorInput
  currency: StringQueryOperatorInput
```

Фича: кодогенерация хуков



Фича: realtime кодогенерация



Фича: моки для тестов

CountriesPage.test.tsx M

src > pages > CountriesPage > CountriesPage.test.tsx > test('shows error indication if request failed') callback

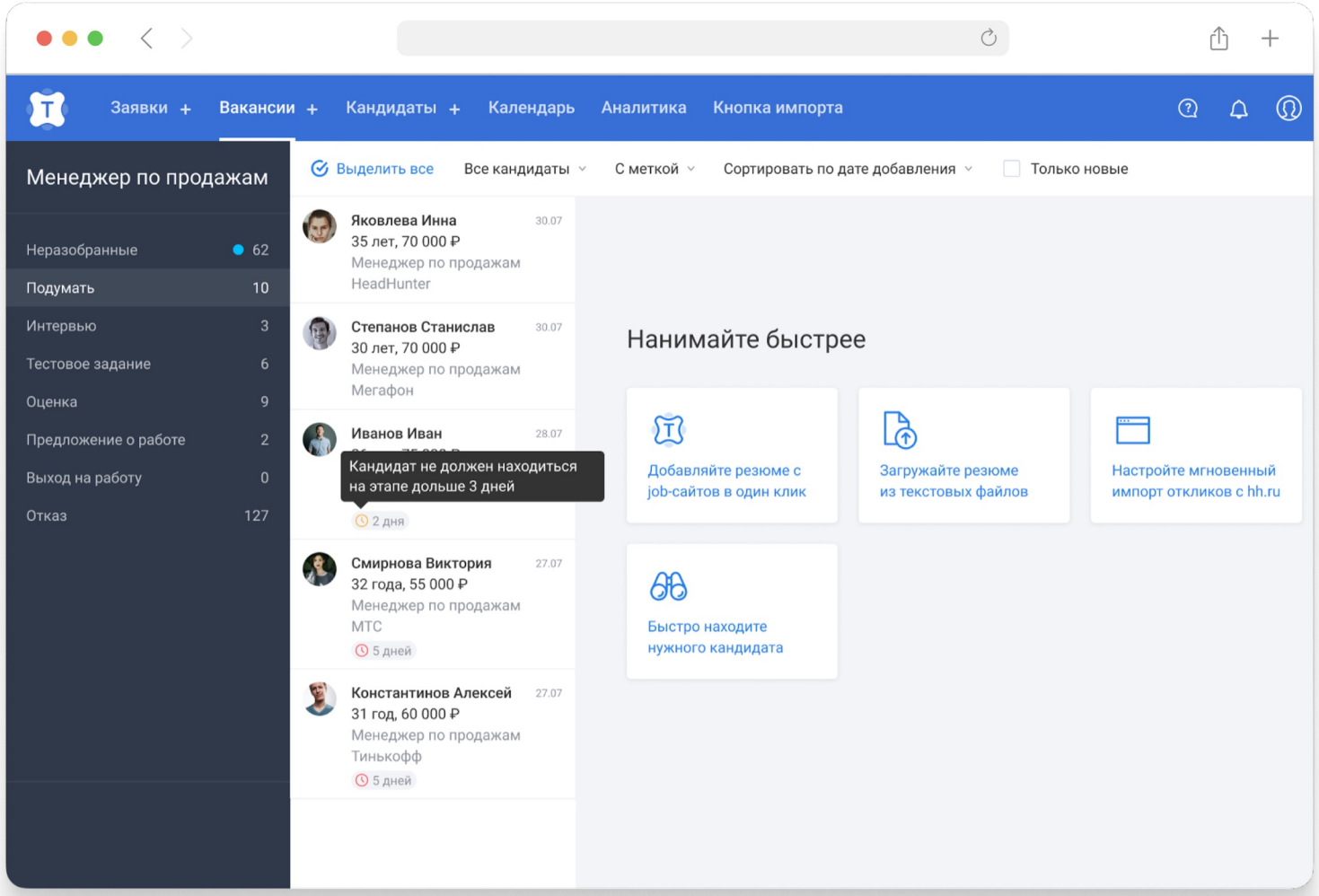
```tsx
import '@testing-library/jest-dom/extend-expect';…

Run | Debug | View story | Profile | Focus
test('shows loading indication while loading', async () => {  51ms
  renderWithProviders(
    <ApolloMockProvider>
      <CountriesPage />
    </ApolloMockProvider>,
  );

  const loadingIndicator = await screen.findByText('Loading...');
  expect(loadingIndicator).toBeDefined();
});

Run | Debug | View story | Profile | Focus
test('shows error indication if request failed', async () => {  41ms
  renderWithProviders(
    <ApolloMockProvider errors={[{ message: 'Fail!' }]}>
      <CountriesPage />
    </ApolloMockProvider>,
  );

  const errorMessage = await screen.findByText('Fail!');
  expect(errorMessage).toBeDefined();
});

Run | Debug | View story | Profile | Focus
test('shows loaded countries', async () => {  41ms
  const queryMocks = {
    countries: () => [{ name: 'Kazakhstan' }, { name: 'Russia' } ],
  };

  renderWithProviders(
    <ApolloMockProvider queryMocks={queryMocks}>
```

Одним словом

Одним словом

Итоги ресерча

Обсуждаем

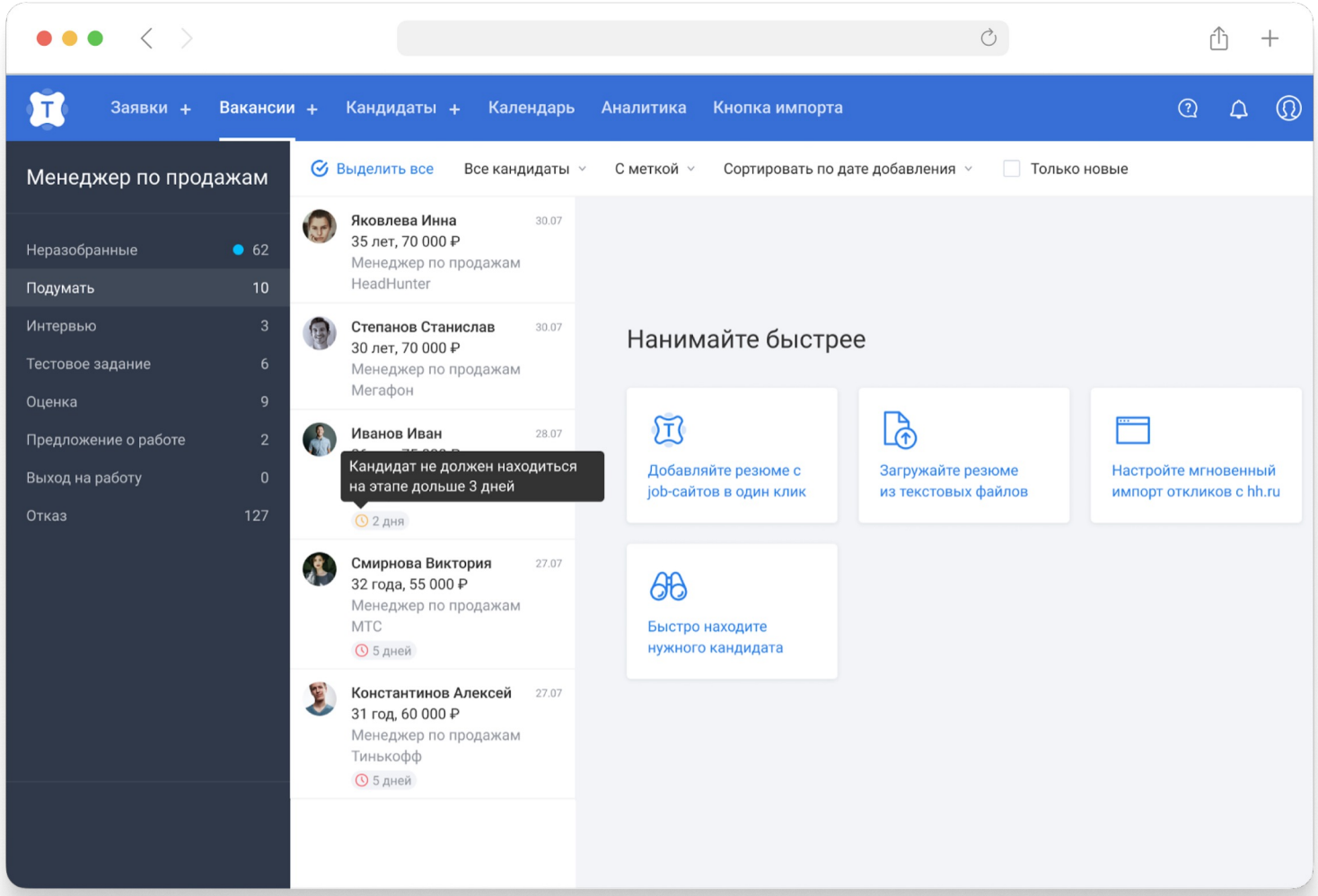

Заявки +
Вакансии +
Кандидаты +
Календарь
Аналитика
Кнопка импорта
Менеджер по продажам
Неразобранные 62
Подумать 10
Интервью 3
Тестовое задание 6
Оценка 9
Предложение о работе 2
Выход на работу 0
Отказ 127
Выделить все
Все кандидаты
С меткой
Сортировать по дате добавления
Только новые
Яковлева Инна 30.07
35 лет, 70 000 ₽
Менеджер по продажам
HeadHunter
Степанов Станислав 30.07
30 лет, 70 000 ₽
Менеджер по продажам
Мегафон
Иванов Иван 28.07
Кандидат не должен находиться на этапе дольше 3 дней
2 дня
Смирнова Виктория 27.07
32 года, 55 000 ₽
Менеджер по продажам
МТС
5 дней
Константинов Алексей 27.07
31 год, 60 000 ₽
Менеджер по продажам
Тинькофф
5 дней
Нанимайте быстрее
Добавляйте резюме с job-сайтов в один клик
Загружайте резюме из текстовых файлов
Настройте мгновенный импорт откликов с hh.ru
Быстро находите нужного кандидата

Обсуждаем

1

Бэк

Говорит, что будем использовать **code-first** подход в генерации схемы, получается высокая оценка в стори поинтах



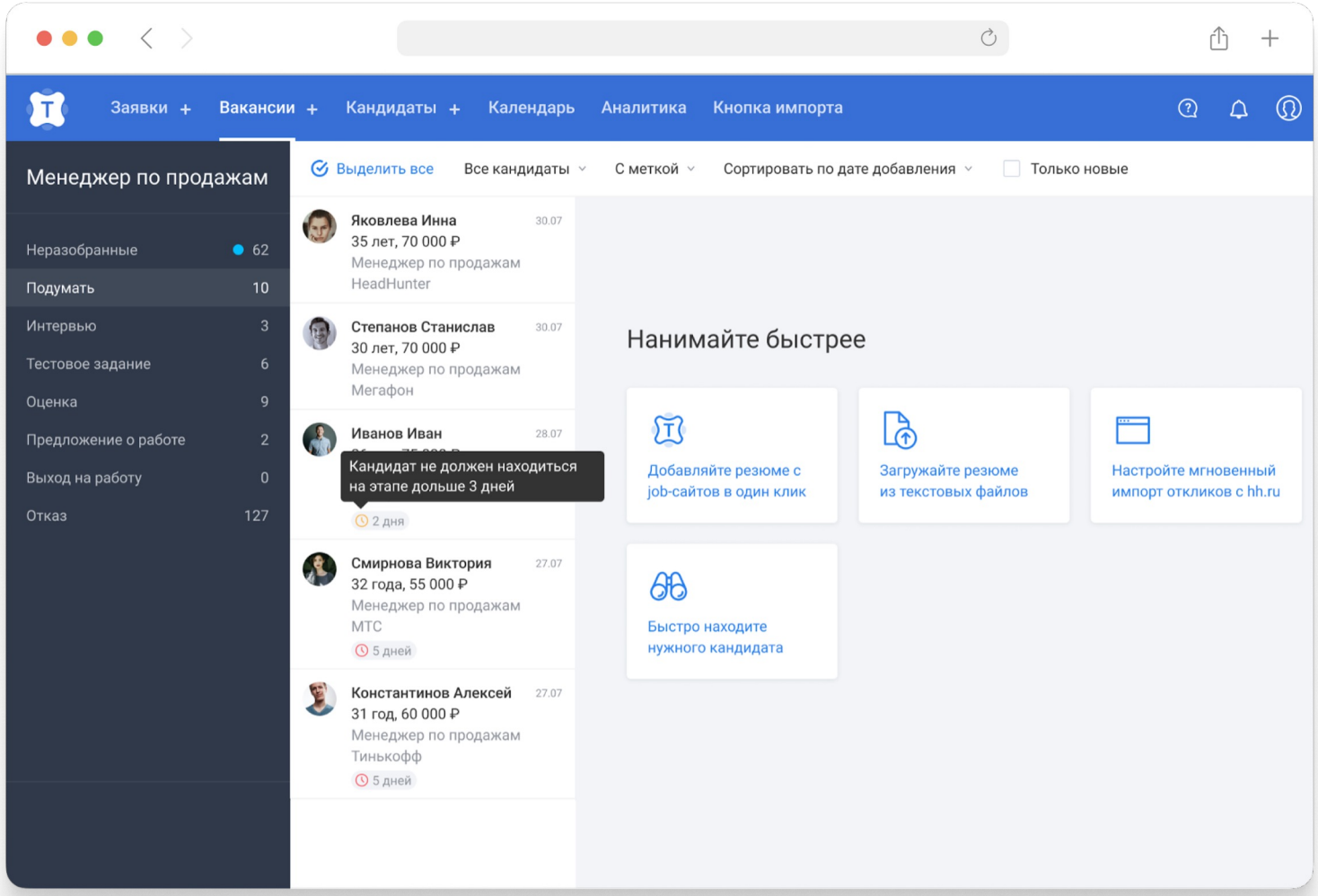
Обсуждаем

1. **Бэк**

   Говорит, что будем использовать **code-first** подход в генерации схемы, получается высокая оценка в стори поинтах

2. **Фронт**

   Докладываю о всей магии.
   Будем пробовать **URQL или Apollo**.
   Выходит оценка ничуть не меньше, чем у бэка



Что входило в frontend оценку?

Что входило в frontend оценку?

- Думать о синхронизации кэша фреймворка с **Redux**

Что входило в frontend оценку?

- Думать о синхронизации кэша фреймворка с **Redux**
- Править в **View** слой компонентов

vacanciesPage.js

app > containers > ats > vacanciesPage.js > AtsVacanciesPage

```
74  class AtsVacanciesPage extends Component {
    tabnine: test | explain | document | ask
102   componentDidMount() {
103     this.loadVacancies();
104   }
105       Nikita Mostovoy, 8 years ago • HH-611488 add my vacancies page
    tabnine: test | explain | document | ask
106   componentDidUpdate(prevProps) {
107     if (
108       this.props[HH_BACKURL_PUBLISHED_PARAMETER] !== prevProps[HH_BACKURL_PUBLISHED_PARAMETER] ||
109       this.props.accountHhVacancyImportSettings !== prevProps.accountHhVacancyImportSettings
110     ) {
111       this.handlePublishedHHId();
112     }
113   }
114
115   loadVacancies = () => {
116     const actualizedFilters = getActualizedFiltersFromLocalStorage(ACTIVE, this.props.vacancies.filters);
117     const actualizedSort = getActualizedSortFromLocalStorage(ACTIVE, this.props.vacancies.sort);
118     this.handlePublishedHHId();
119     this.props.loadVacancies(ACTIVE, ACTIVE_VACANCIES_URL, actualizedFilters, actualizedSort);
120     this.props.resetSelectedPersons();
121   };
122
123   checkSelectable = (vacancy) =>
124     hasAllOfPermissions({ context: { name: 'vacancy', id: vacancy.id } }, CHANGE_VACANCY_STATUS);
125
126   openApplyTemplateToVacancies = (vacancyIdsToApplyTemplate) => {
127     const dispatch = (action) => getOrCreateStore().dispatch(action);
128     const actualizedFilters = getActualizedFiltersFromLocalStorage(ACTIVE, this.props.vacancies.filters);
129     const actualizedSort = getActualizedSortFromLocalStorage(ACTIVE, this.props.vacancies.sort);
130
131     dispatch(loadLockedVacancies(ACTIVE_VACANCIES_URL, actualizedFilters, actualizedSort)).then((data) => {
132       const vacancyIdsLocked = data.data.ats_vacancies_active_items
```

HH-209541 3↓ 0↑ ⊗ 1K ⚠ 0 ⓘ 28 0 tms-static SFTP: Profile: TMS Pull Request #5261 Ln 105, Col 1 Spaces: 2 UTF-8 LF {} JavaS

Что входило в frontend оценку?

- Думать о синхронизации кэша фреймворка с **Redux**
- Править в **View** слой компонентов
- Переписывать **React.Class** на React.**FC**

vacanciesPage.js

app > containers > ats > vacanciesPage.js > AtsVacanciesPage

```
74  class AtsVacanciesPage extends Component {
    tabnine: test | explain | document | ask
102   componentDidMount() {
103     this.loadVacancies();
104   }
105       Nikita Mostovoy, 8 years ago • HH-611488 add my vacancies page
    tabnine: test | explain | document | ask
106   componentDidUpdate(prevProps) {
107     if (
108       this.props[HH_BACKURL_PUBLISHED_PARAMETER] !== prevProps[HH_BACKURL_PUBLISHED_PARAMETER] ||
109       this.props.accountHhVacancyImportSettings !== prevProps.accountHhVacancyImportSettings
110     ) {
111       this.handlePublishedHHId();
112     }
113   }
114
115   loadVacancies = () => {
116     const actualizedFilters = getActualizedFiltersFromLocalStorage(ACTIVE, this.props.vacancies.filters);
117     const actualizedSort = getActualizedSortFromLocalStorage(ACTIVE, this.props.vacancies.sort);
118     this.handlePublishedHHId();
119     this.props.loadVacancies(ACTIVE, ACTIVE_VACANCIES_URL, actualizedFilters, actualizedSort);
120     this.props.resetSelectedPersons();
121   };
122
123   checkSelectable = (vacancy) =>
124     hasAllOfPermissions({ context: { name: 'vacancy', id: vacancy.id } }, CHANGE_VACANCY_STATUS);
125
126   openApplyTemplateToVacancies = (vacancyIdsToApplyTemplate) => {
127     const dispatch = (action) => getOrCreateStore().dispatch(action);
128     const actualizedFilters = getActualizedFiltersFromLocalStorage(ACTIVE, this.props.vacancies.filters);
129     const actualizedSort = getActualizedSortFromLocalStorage(ACTIVE, this.props.vacancies.sort);
130
131     dispatch(loadLockedVacancies(ACTIVE_VACANCIES_URL, actualizedFilters, actualizedSort)).then((data) => {
132       const vacancyIdsLocked = data.data.ats.vacancies.active.items
```

HH-209541 3↓ 0↑ ⊗ 1K ⚠ 0 ⓘ 28 0 tms-static SFTP: Profile: TMS Pull Request #5261 Ln 105, Col 1 Spaces: 2 UTF-8 LF {} JavaS

Что входило в frontend оценку?

- Думать о синхронизации кэша фреймворка с **Redux**
- Править в **View** слой компонентов
- Переписывать **React.Class** на React.**FC**
- Обучить **новой** технологии и **фреймворку**

vacanciesPage.js

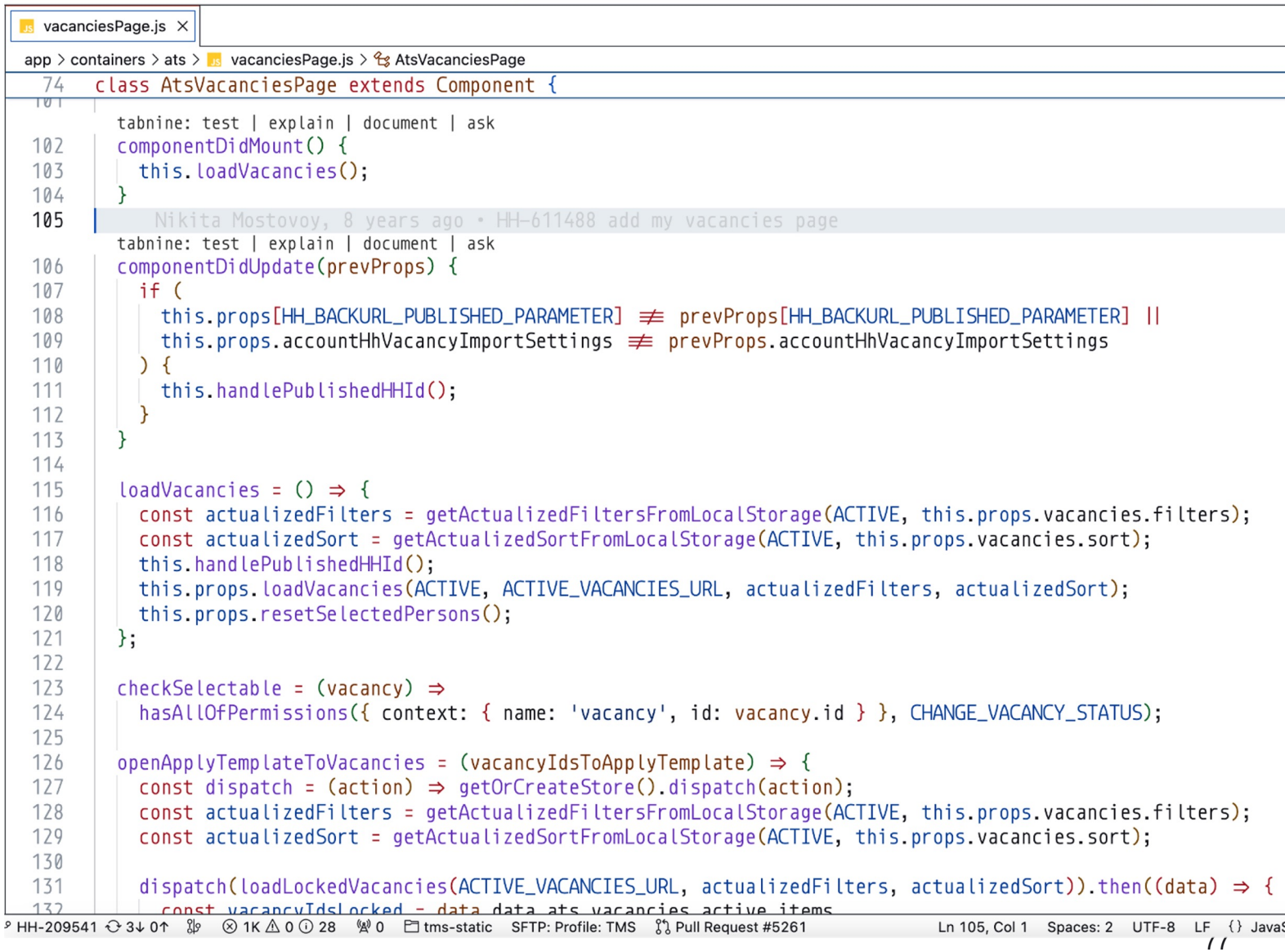
app > containers > ats > vacanciesPage.js > AtsVacanciesPage

```
74  class AtsVacanciesPage extends Component {
    tabnine: test | explain | document | ask
102   componentDidMount() {
103     this.loadVacancies();
104   }
105       Nikita Mostovoy, 8 years ago • HH-611488 add my vacancies page
    tabnine: test | explain | document | ask
106   componentDidUpdate(prevProps) {
107     if (
108       this.props[HH_BACKURL_PUBLISHED_PARAMETER] !== prevProps[HH_BACKURL_PUBLISHED_PARAMETER] ||
109       this.props.accountHhVacancyImportSettings !== prevProps.accountHhVacancyImportSettings
110     ) {
111       this.handlePublishedHHId();
112     }
113   }
114
115   loadVacancies = () => {
116     const actualizedFilters = getActualizedFiltersFromLocalStorage(ACTIVE, this.props.vacancies.filters);
117     const actualizedSort = getActualizedSortFromLocalStorage(ACTIVE, this.props.vacancies.sort);
118     this.handlePublishedHHId();
119     this.props.loadVacancies(ACTIVE, ACTIVE_VACANCIES_URL, actualizedFilters, actualizedSort);
120     this.props.resetSelectedPersons();
121   };
122
123   checkSelectable = (vacancy) =>
124     hasAllOfPermissions({ context: { name: 'vacancy', id: vacancy.id } }, CHANGE_VACANCY_STATUS);
125
126   openApplyTemplateToVacancies = (vacancyIdsToApplyTemplate) => {
127     const dispatch = (action) => getOrCreateStore().dispatch(action);
128     const actualizedFilters = getActualizedFiltersFromLocalStorage(ACTIVE, this.props.vacancies.filters);
129     const actualizedSort = getActualizedSortFromLocalStorage(ACTIVE, this.props.vacancies.sort);
130
131     dispatch(loadLockedVacancies(ACTIVE_VACANCIES_URL, actualizedFilters, actualizedSort)).then((data) => {
132       const vacancyIdsLocked = data.data.ats.vacancies.active.items
```

HH-209541 3↓ 0↑ ⊗ 1K ⚠ 0 ⓘ 28 0 tms-static SFTP: Profile: TMS Pull Request #5261 Ln 105, Col 1 Spaces: 2 UTF-8 LF {} JavaS

Что входило в frontend оценку?

- Думать о синхронизации кэша фреймворка с **Redux**
- Править в **View** слой компонентов
- Переписывать **React.Class** на React.**FC**
- Обучить **новой** технологии и **фреймворку**
- Работать с **Interceptors**



Оценка: больше одного месяца с обеих сторон

Все, что могли — отрезали

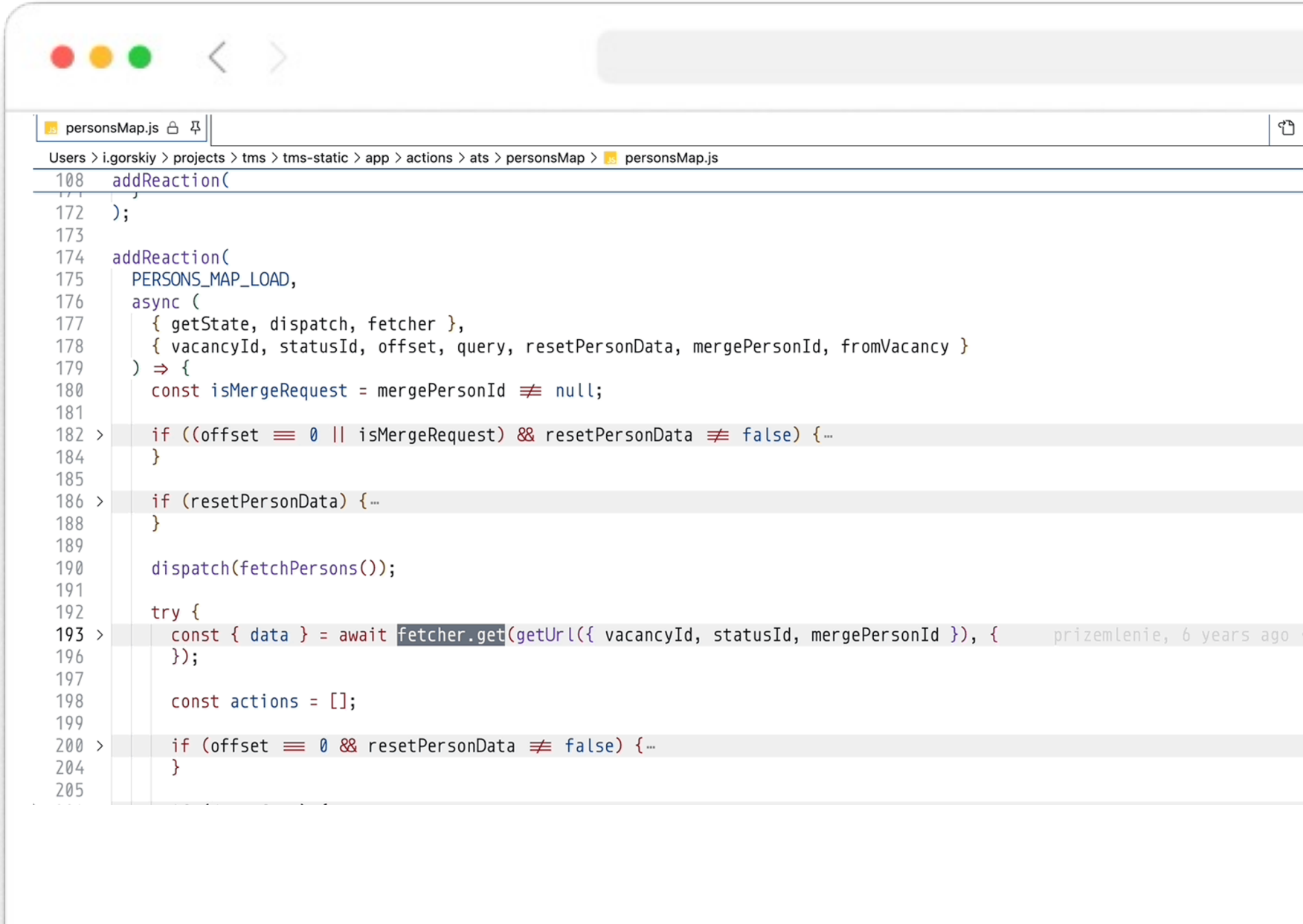
Все, что могли — отрезали

Челлендж

Челлендж

Помните асинхронные экшены?

Помните асинхронные экшены?



Напишем свой fetcher и подменим его!

Важно!

Работаем только на уровне сети

Важно!

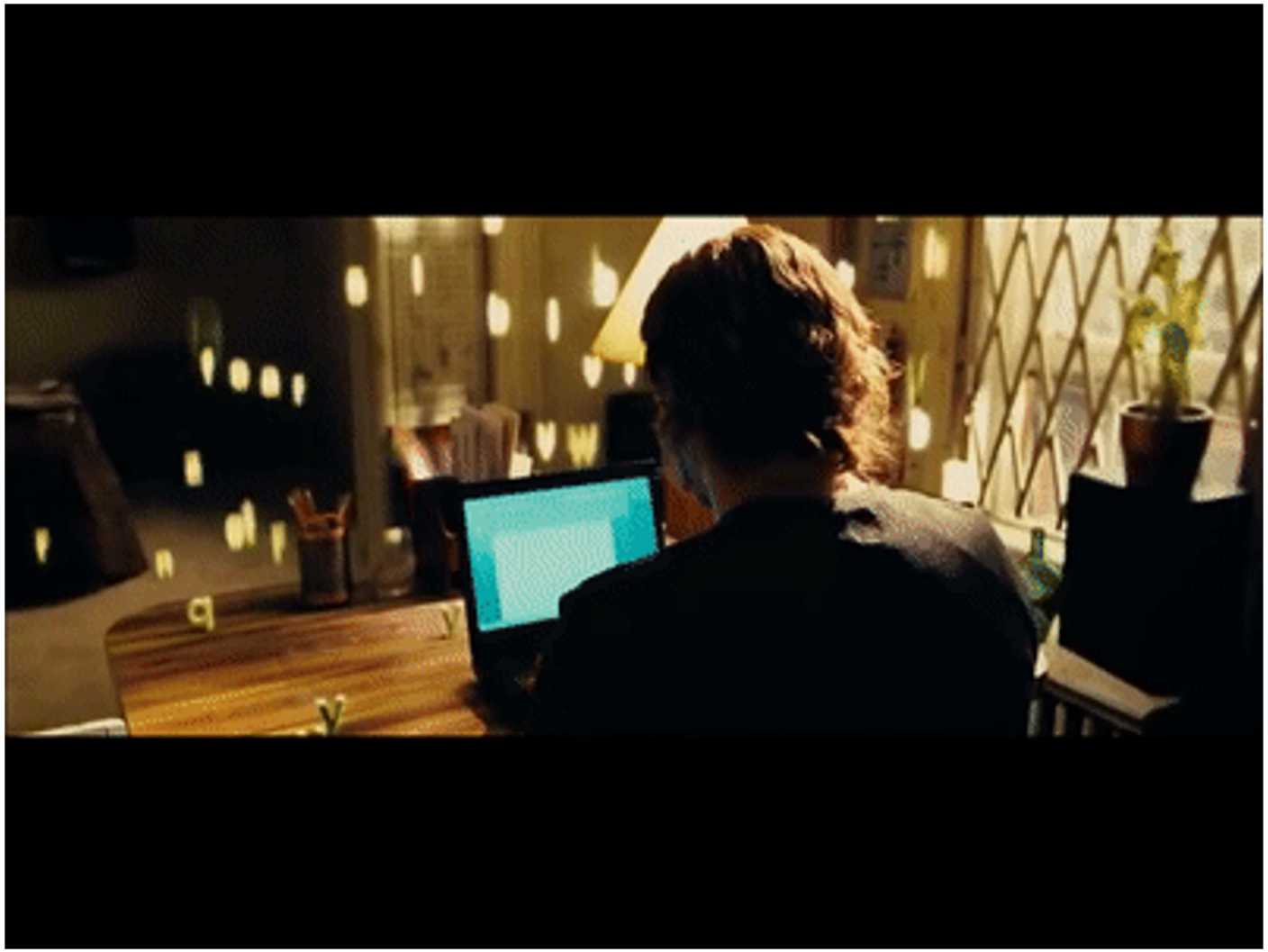
Не трогаем слой view

Получается сильно дешевле внедрения фреймворка

Мысль

Сила фреймворков построена на туллинге, без него не имеет смысла брать GraphQL в прод

Погнали



Тулинг

01.

Клиент

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

02.

Плейграунд

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

02.

Плейграунд

- Доступная песочница для выполнения запросов и мутаций к нашему бэкенду, аналогично фреймворкам

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

02.

Плейграунд

- Доступная песочница для выполнения запросов и мутаций к нашему бэкенду, аналогично фреймворкам
- Кастомизация под наш продукт

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

02.

Плейграунд

- Доступная песочница для выполнения запросов и мутаций к нашему бэкенду, аналогично фреймворкам
- Кастомизация под наш продукт

03.

Кодогенерация

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

02.

Плейграунд

- Доступная песочница для выполнения запросов и мутаций к нашему бэкенду, аналогично фреймворкам
- Кастомизация под наш продукт

03.

Кодогенерация

- Получение схемы

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

02.

Плейграунд

- Доступная песочница для выполнения запросов и мутаций к нашему бэкенду, аналогично фреймворкам
- Кастомизация под наш продукт

03.

Кодогенерация

- Получение схемы
- Генерация типов по схеме

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

02.

Плейграунд

- Доступная песочница для выполнения запросов и мутаций к нашему бэкенду, аналогично фреймворкам
- Кастомизация под наш продукт

03.

Кодогенерация

- Получение схемы
- Генерация типов по схеме
- Генерация fetcher функций под каждый запрос/мутацию

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

02.

Плейграунд

- Доступная песочница для выполнения запросов и мутаций к нашему бэкенду, аналогично фреймворкам
- Кастомизация под наш продукт

03.

Кодогенерация

- Получение схемы
- Генерация типов по схеме
- Генерация fetcher функций под каждый запрос/мутацию
- Генерация фрагментов

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

02.

Плейграунд

- Доступная песочница для выполнения запросов и мутаций к нашему бэкенду, аналогично фреймворкам
- Кастомизация под наш продукт

03.

Кодогенерация

- Получение схемы
- Генерация типов по схеме
- Генерация fetcher функций под каждый запрос/мутацию
- Генерация фрагментов

04.

Тесты

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

02.

Плейграунд

- Доступная песочница для выполнения запросов и мутаций к нашему бэкенду, аналогично фреймворкам
- Кастомизация под наш продукт

03.

Кодогенерация

- Получение схемы
- Генерация типов по схеме
- Генерация fetcher функций под каждый запрос/мутацию
- Генерация фрагментов

04.

Тесты

- Мок операций

Тулинг

01.

Клиент

- Умеет работать с GraphQL файлами и запросами
- Поддерживает все интерцепторы нашего проекта
- Поддерживает методы текущего клиента
- Отладка и обработка ошибок

02.

Плейграунд

- Доступная песочница для выполнения запросов и мутаций к нашему бэкенду, аналогично фреймворкам
- Кастомизация под наш продукт

03.

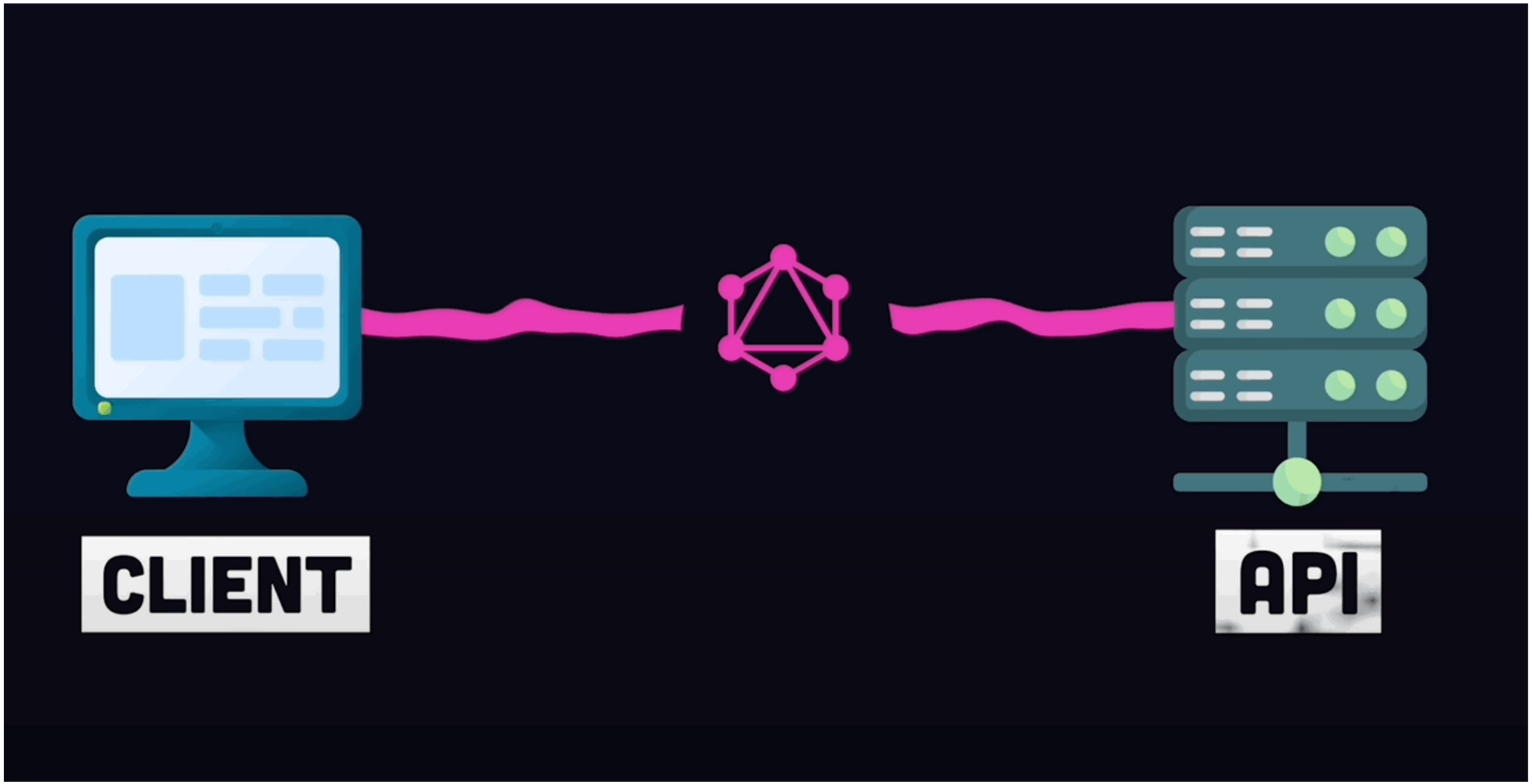
Кодогенерация

- Получение схемы
- Генерация типов по схеме
- Генерация fetcher функций под каждый запрос/мутацию
- Генерация фрагментов

04.

Тесты

- Мок операций
- Генерация стабов для тестов


CLIENT
API

Клиент: контракт GraphQL response

```ts
/**
 * @see https://spec.graphql.org/October2021/#sec-Errors.Error-result-format
 */
interface Error {
    message: string
    column: { line: number; column: number }[]
    path: string[]
    [x: string]: unknown
}
/**
 * @see https://spec.graphql.org/October2021/#sec-Errors
 */
interface FailResponse<T> {
    data: T | null
    errors: Error[]
}
/**
 * @see https://spec.graphql.org/October2021/#sec-Response
 */
interface SuccessResponse<T> {
    data: T
    errors: []
}

export type Response<T extends object> = SuccessResponse<T> | FailResponse<T>
```

Клиент: контракт GraphQL response

```ts
/**
 * @see https://spec.graphql.org/October2021/#sec-Errors.Error-result-format
 */
interface Error {
    message: string
    column: { line: number; column: number }[]
    path: string[]
    [x: string]: unknown
}
/**
 * @see https://spec.graphql.org/October2021/#sec-Errors
 */
interface FailResponse<T> {
    data: T | null
    errors: Error[]
}
/**
 * @see https://spec.graphql.org/October2021/#sec-Response
 */
interface SuccessResponse<T> {
    data: T
    errors: []
}

export type Response<T extends object> = SuccessResponse<T> | FailResponse<T>
```

Клиент: request

```ts
import { type DocumentNode } from 'graphql'

export type Query = DocumentNode | string

export type Variables<T extends object> = T

export interface RequestFn {
    <D extends object, V extends object>(query: Query variables?: Variables<V>): Promise<Response<D>>
}
```

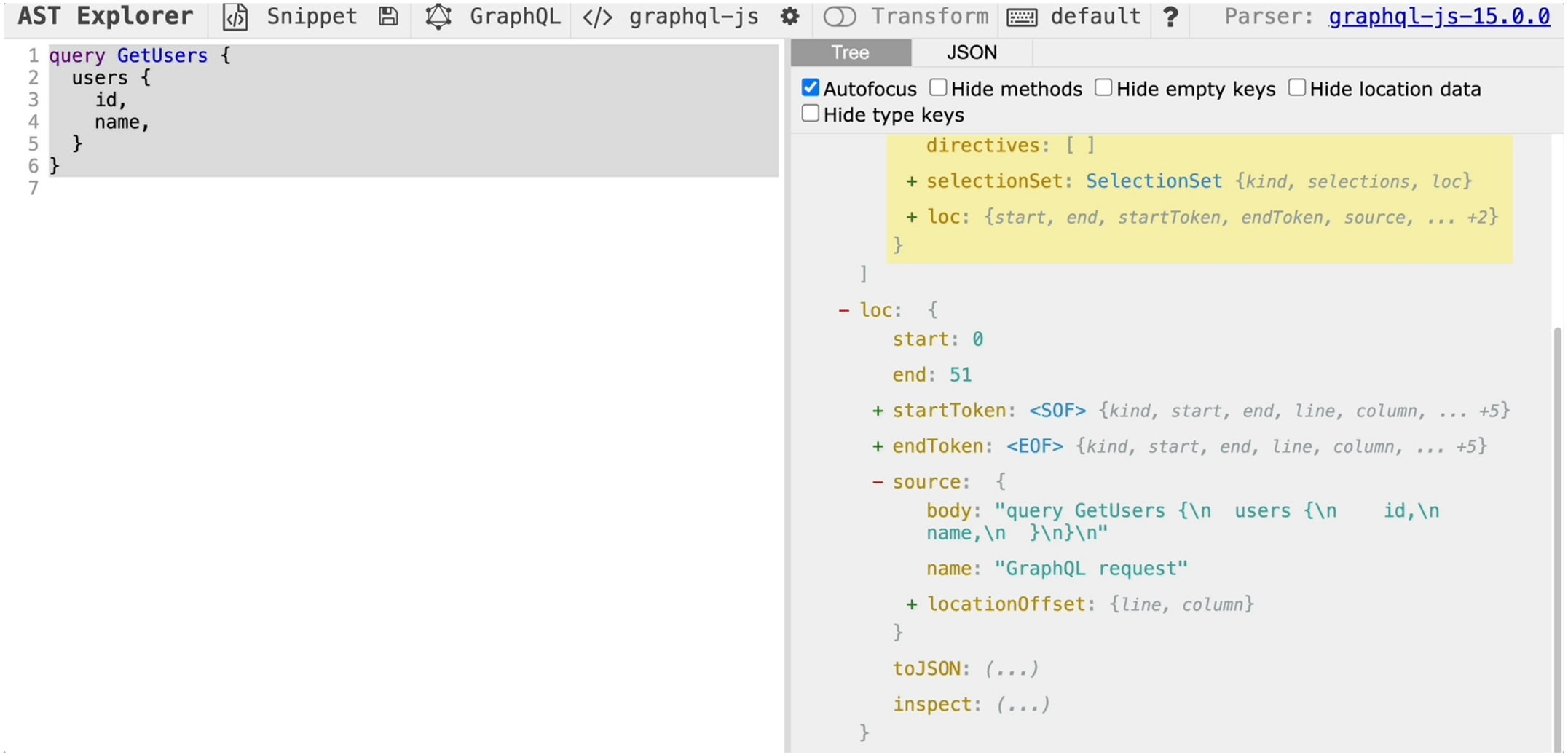
Клиент: request

```ts
import { type ASTNode, type OperationDefinitionNode } from 'graphql'
import type { RequestFn } from './types'

const isOperation = (node: ASTNode): node is OperationDefinitionNode => node.kind === 'OperationDefinition'
const getOperationName = (node: ASTNode): string => isOperation(node) ? node.name!.value : 'unknown'

export const request: <U extends string>(url: U) => RequestFn = (url) => async (query, variables) => {
    const isDocumentNode = typeof query !== 'string';
    const operationName = isDocumentNode ? getOperationName(query.definitions[0]) : 'unknown';
    const api = new URL(url)
    const params = {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            query:isDocumentNode ?query.loc?.source.body : query,
            variables,
            operationName,
        }),
    }
    api.searchParams.append('operationName',operationName)

    const response = await fetch(`${api}`, params)
    const data = await response.json()

    return data
}
```

Клиент: request

```ts
import { type ASTNode, type OperationDefinitionNode } from 'graphql'
import type { RequestFn } from './types'

const isOperation = (node: ASTNode): node is OperationDefinitionNode => node.kind === 'OperationDefinition'
const getOperationName = (node: ASTNode): string => isOperation(node) ? node.name!.value : 'unknown'

export const request: <U extends string>(url: U) => RequestFn = (url) => async (query, variables) => {
    const isDocumentNode = typeof query !== 'string';
    const operationName = isDocumentNode ? getOperationName(query.definitions[0]) : 'unknown';
    const api = new URL(url)
    const params = {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            query:isDocumentNode ? query.loc?.source.body : query,
            variables,
            operationName,
        }),
    }
    api.searchParams.append('operationName',operationName)

    const response = await fetch(`${api}`, params)
    const data = await response.json()

    return data
}
```

Клиент: AST DocumentNode

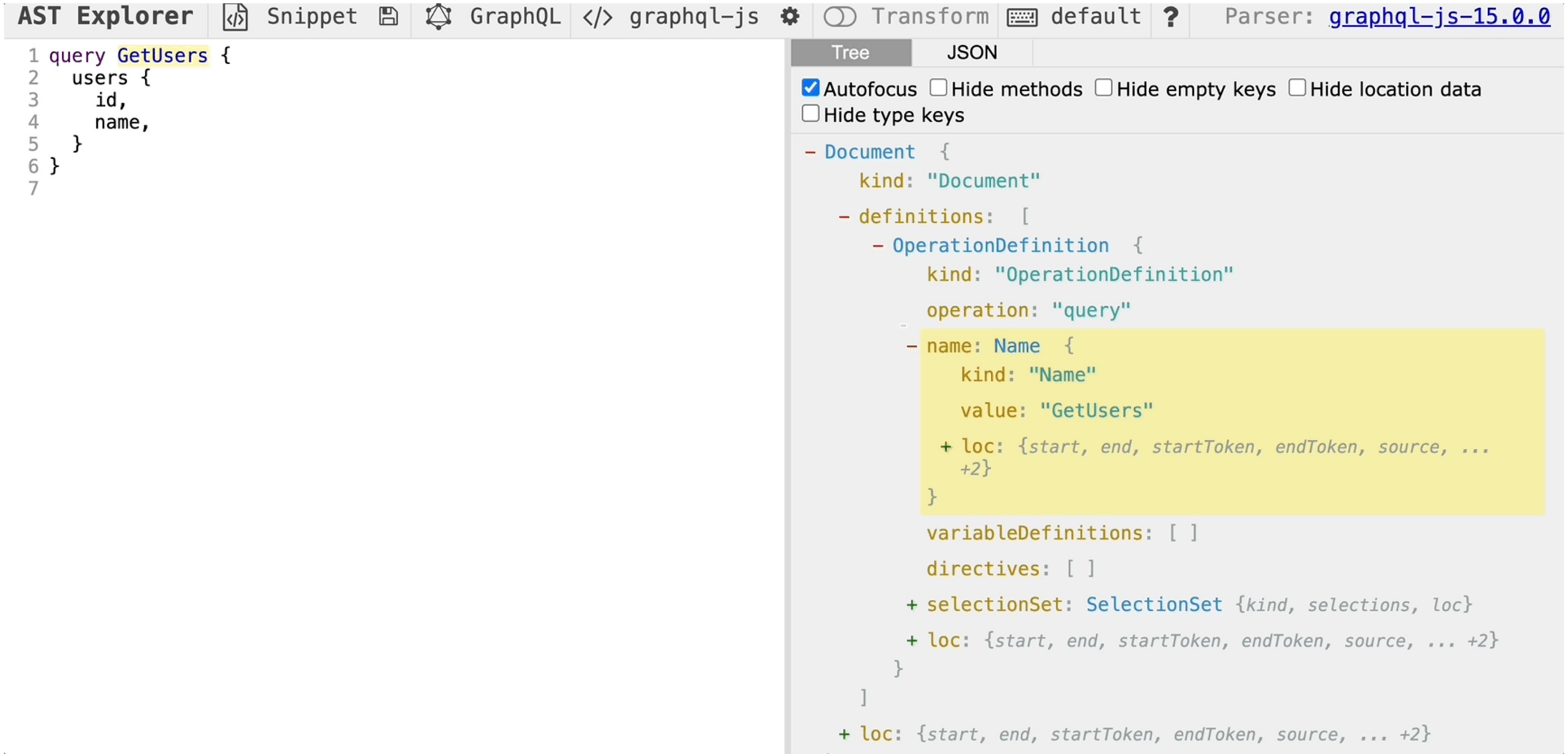


Клиент: operationName ключ к успеху

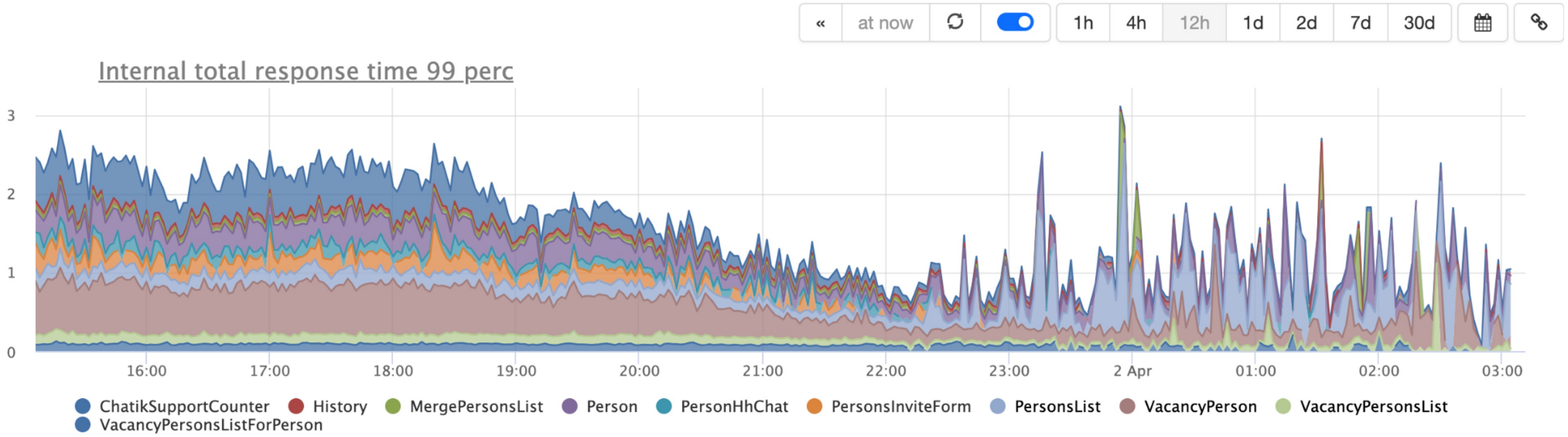
Клиент: добавляем operationName

```ts
import { type ASTNode, type OperationDefinitionNode } from 'graphql'
import type { RequestFn } from './types'

const isOperation = (node: ASTNode): node is OperationDefinitionNode => node.kind === 'OperationDefinition'
const getOperationName = (node: ASTNode): string => isOperation(node) ? node.name!.value : 'unknown'

export const request: <U extends string>(url: U) => RequestFn = (url) => async (query, variables) => {
    const isDocumentNode = typeof query !== 'string';
    const operationName = isDocumentNode ? getOperationName(query.definitions[0]) : 'unknown';
    const api = new URL(url)
    const params = {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            query:isDocumentNode ? query.loc?.source.body : query,
            variables,
            operationName,
        }),
    }
    api.searchParams.append('operationName',operationName)

    const response = await fetch(`${api}`, params)
    const data = await response.json()

    return data
}
```

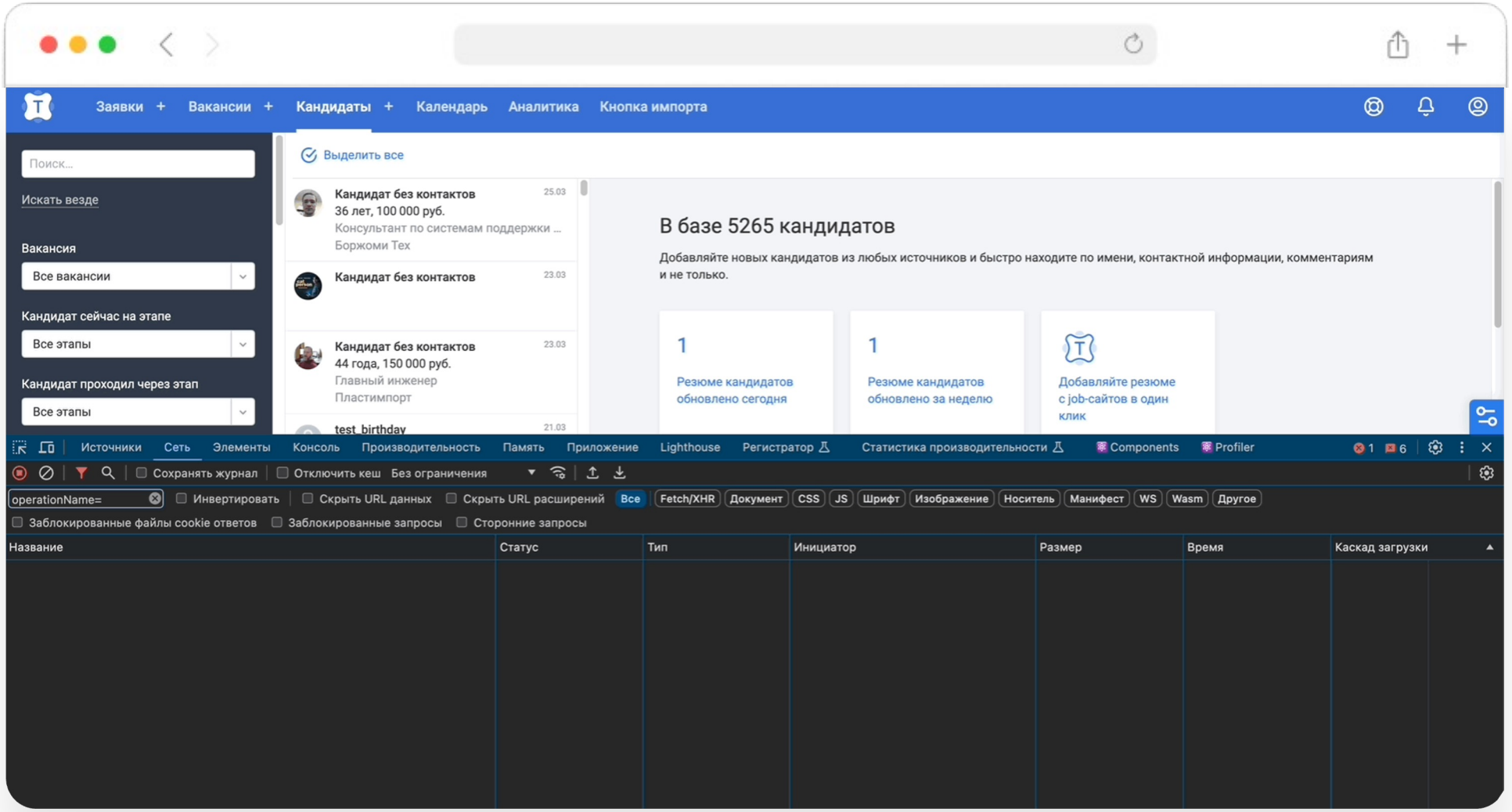
Клиент: AST operationName



Клиент: operationName для мониторинга



Клиент: operationName в URL для отладки



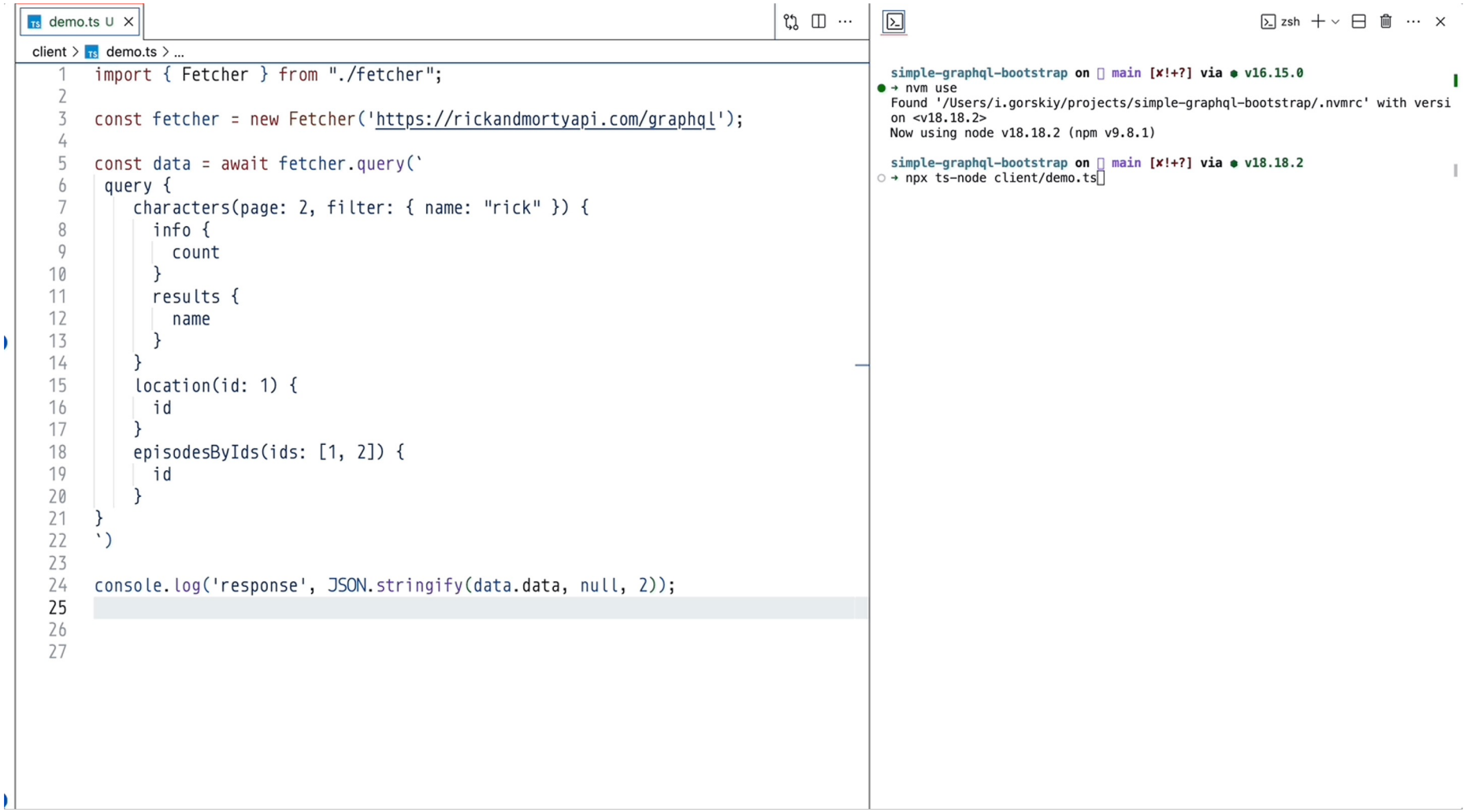
Клиент готов

```
1  import type { Query, Variables } from './types'
2  import { request } from './request'
3
4  export class Fetcher {
5      public origin: string
6      public url: string
7
8      constructor(url: string) {
9          const graphql = new URL(url)
10         this.url = graphql.toString()
11         this.origin = graphql.origin
12     }
13
14     query<D extends object, V extends object = object>(q:Query, v?:Variables<V>) {
15         return request(this.url)<D,V>(q, v)
16     }
17
18     mutation<D extends object, V extends object = object>(q:Query, v?:Variables<V>) {
19         return request(this.url)<D,V>(q, v)
20     }
21 }
```

Клиент готов

```
1  import type { Query, Variables } from './types'
2  import { request } from './request'
3
4  export class Fetcher {
5      public origin: string
6      public url: string
7
8      constructor(url: string) {
9          const graphql = new URL(url)
10         this.url = graphql.toString()
11         this.origin = graphql.origin
12     }
13
14     query<D extends object, V extends object = object>(q:Query, v?:Variables<V>) {
15         return request(this.url)<D,V>(q, v)
16     }
17
18     mutation<D extends object, V extends object = object>(q:Query, v?:Variables<V>) {
19         return request(this.url)<D,V>(q, v)
20     }
21 }
```

Проверяем


demo.ts U
client > demo.ts > ...
1 import { Fetcher } from "./fetcher";
2
3 const fetcher = new Fetcher('https://rickandmortyapi.com/graphql');
4
5 const data = await fetcher.query(`
6 query {
7 characters(page: 2, filter: { name: "rick" }) {
8 info {
9 count
10 }
11 results {
12 name
13 }
14 }
15 location(id: 1) {
16 id
17 }
18 episodesByIds(ids: [1, 2]) {
19 id
20 }
21 }
22 `)
23
24 console.log('response', JSON.stringify(data.data, null, 2));
25
26
27
zsh
simple-graphql-bootstrap on main [x!+?] via v16.15.0
➜ nvm use
Found '/Users/i.gorskiy/projects/simple-graphql-bootstrap/.nvmrc' with versi
on <v18.18.2>
Now using node v18.18.2 (npm v9.8.1)
simple-graphql-bootstrap on main [x!+?] via v18.18.2
➜ npx ts-node client/demo.ts

40 строчек кода и клиент готов

Плейграунд

Кодогенерация

Кодогенерация: .graphqlrc.yaml

```yaml
schema: ".codegen/schema.graphql"
documents:
  - "./src/**/*.graphql"
```

Кодогенерация: .graphqlrc.yaml

```
1  schema: ".codegen/schema.graphql"
2  documents:
3    - "./src/**/*.graphql"
```

Кодогенерация: .graphqlrc.yaml

```
1  schema: ".codegen/schema.graphql"
2  documents:
3    - "./src/**/*.graphql"
```

Кодогенерация: получаем schema.GraphQL

```ts
import path from 'node:path'
import fs from 'node:fs'
import {
    getIntrospectionQuery,
    buildClientSchema,
    IntrospectionQuery,
} from 'graphql'
import { Fetcher } from '../../client/fetcher'

async function fetchSchema(url: string) {
    const fetcher = new Fetcher(url)
    const { data } = await fetcher.query<IntrospectionQuery>(
        getIntrospectionQuery()
    )

    if (data) {
        console.log('✅', 'download schema introspection')
        return buildClientSchema(data)
    }
}
```

Кодогенерация: получаем schema.graphql

```ts
import {
    getIntrospectionQuery,
    buildClientSchema,
    IntrospectionQuery,
} from 'graphql'
import { Fetcher } from '../../client/fetcher'

async function fetchSchema(url: string) {
    const fetcher = new Fetcher(url)
    const { data } = await fetcher.query<IntrospectionQuery>(
        getIntrospectionQuery()
    )

    if (data) {
        console.log('✅', 'download schema introspection')
        return buildClientSchema(data)
    }
}
```

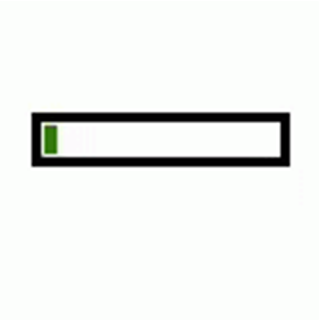
Кодогенерация: получаем schema.GraphQL

```
function printClientSchema(data: GraphQLSchema): string {
    const schema = printSchema(data)
    console.log('✅', 'print client schema')
    return schema
}

function writeSchema(dist: string, schema: string) {
    const output = path.resolve(process.cwd(), dist)
    console.log('✅', 'write schema file', dist)
    fs.writeFileSync(output, schema)
}

function printFile(path: string) {
    return (schema?: GraphQLSchema) => {
        !fs.existsSync(path) && fs.mkdirSync(path)
        schema && writeSchema(`${path}/schema.graphql`, printClientSchema(schema))
    }
}

fetchSchema('https://rickandmortyapi.com/graphql')
    .then(printFile('./codegen'))
    .then(() => console.log('✅', 'success create schema.graphql'))
    .catch((e) => console.error('❌', 'failed create schema.graphql', 'error:', e))
```

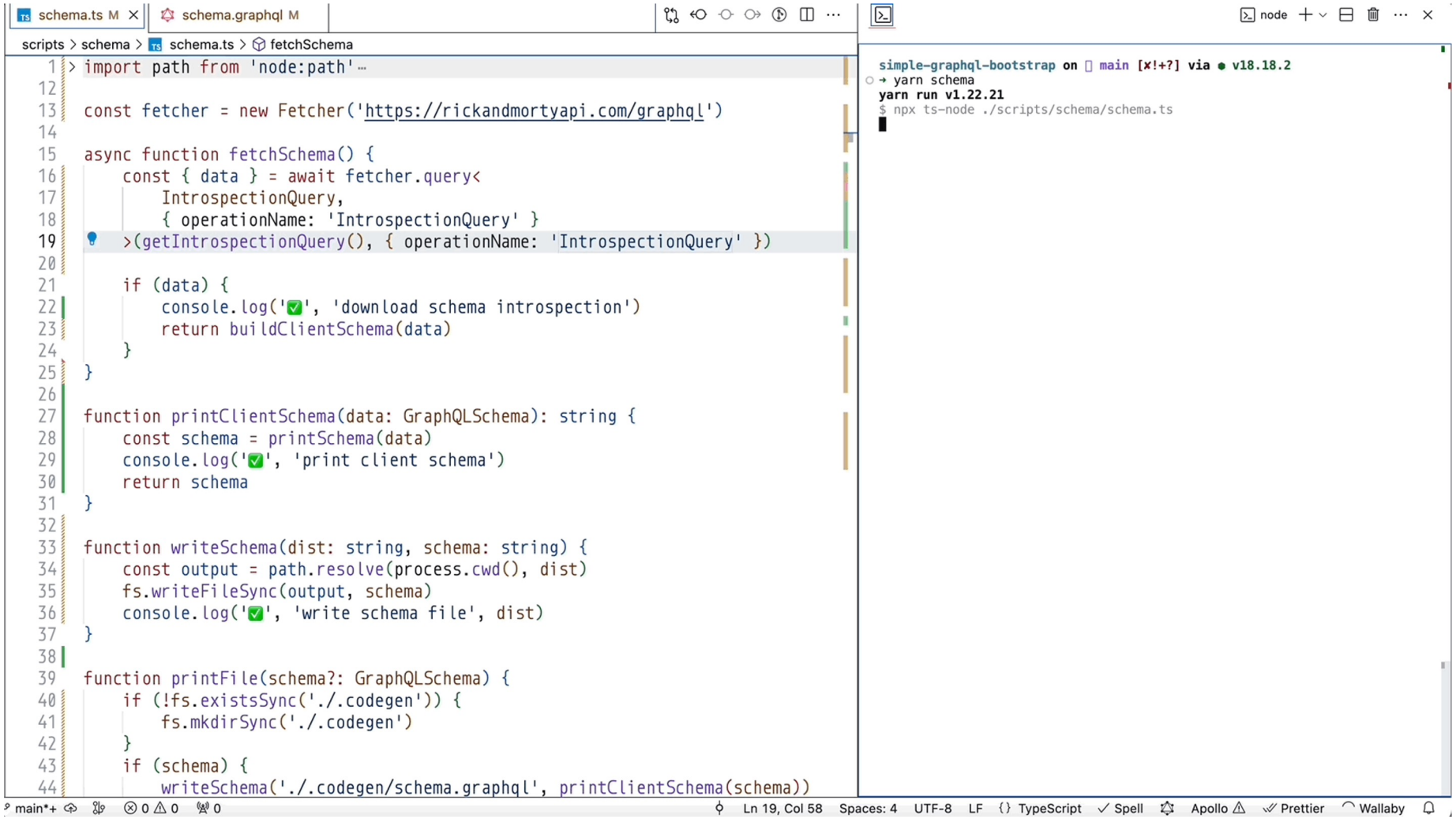
Кодогенерация: получаем schema.graphql

```
function printClientSchema(data: GraphQLSchema): string {
    const schema = printSchema(data)
    console.log('✅', 'print client schema')
    return schema
}

function writeSchema(dist: string, schema: string) {
    const output = path.resolve(process.cwd(), dist)
    console.log('✅', 'write schema file', dist)
    fs.writeFileSync(output, schema)
}

function printFile(path: string) {
    return (schema?: GraphQLSchema) => {
        !fs.existsSync(path) && fs.mkdirSync(path)
        schema && writeSchema(`${path}/schema.graphql`, printClientSchema(schema))
    }
}

fetchSchema('https://rickandmortyapi.com/graphql')
    .then(printFile('./codegen'))
    .then(() => console.log('✅', 'success create schema.graphql'))
    .catch((e) => console.error('❌', 'failed create schema.graphql', 'error:', e))
```

Кодогенерация: получаем schema.graphql



Кодогенерация: получаем schema.graphql

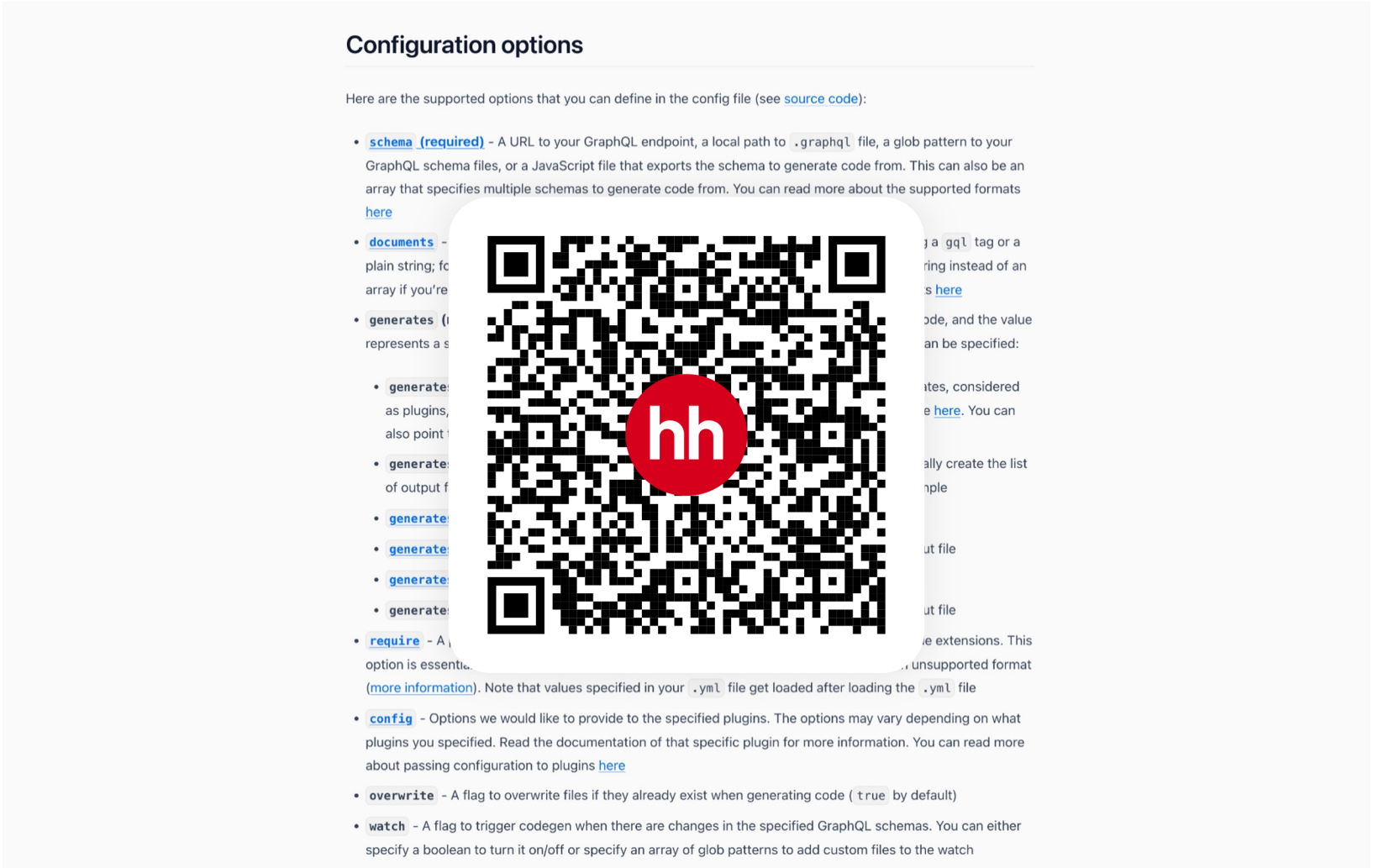


Кодогенерация: продолжаем

Кодогенерация: The Guild

GraphQL codegen

Кодогенерация: graphql-codegen



Configuration options

Here are the supported options that you can define in the config file (see source code):

- `schema` **(required)** - A URL to your GraphQL endpoint, a local path to `.graphql` file, a glob pattern to your GraphQL schema files, or a JavaScript file that exports the schema to generate code from. This can also be an array that specifies multiple schemas to generate code from. You can read more about the supported formats here
- `documents` -
- `generates` (
- `require` - A
- `config` - Options we would like to provide to the specified plugins. The options may vary depending on what plugins you specified. Read the documentation of that specific plugin for more information. You can read more about passing configuration to plugins here
- `overwrite` - A flag to overwrite files if they already exist when generating code (`true` by default)
- `watch` - A flag to trigger codegen when there are changes in the specified GraphQL schemas. You can either specify a boolean to turn it on/off or specify an array of glob patterns to add custom files to the watch

Кодогенерация: graphql-codegen

```
yarn add --dev typescript @graphql-codegen/cli
yarn add --dev @graphql-codegen/typescript
```

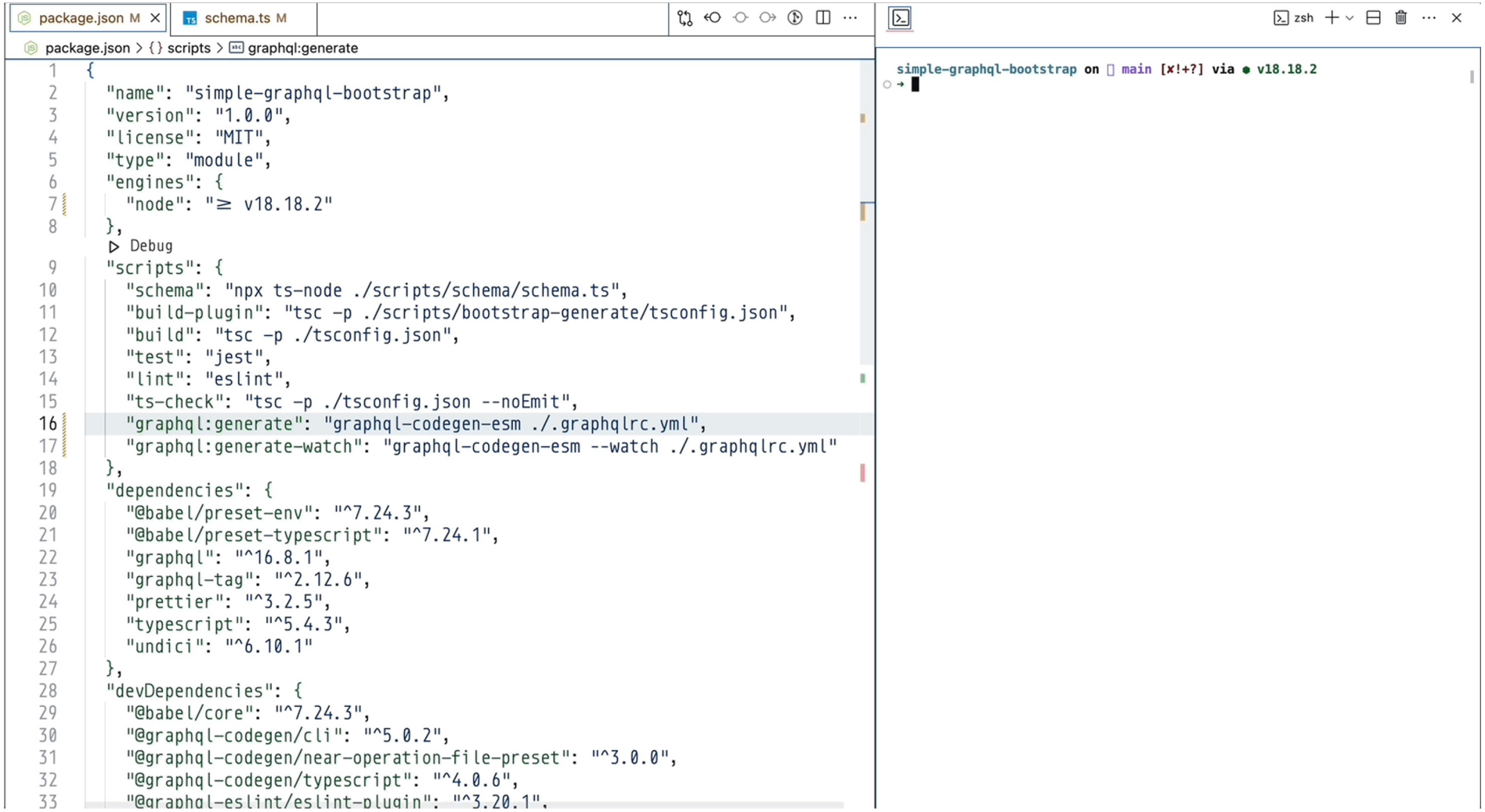
Кодогенерация: добавляем типизированную схему

```yaml
schema: ".codegen/schema.graphql"
documents:
  - "./src/**/*.graphql"
ignoreNoDocuments: true
extensions:
  codegen:
    errorsOnly: true
    generates:
      ./types/schema.ts:
        config:
          defaultScalarType: unknown
          scalars:
            ID: "number"
        plugins:
          - typescript
          - add:
              content: "/* eslint-disable */\n \n /* Auto generated file */ \n "
```

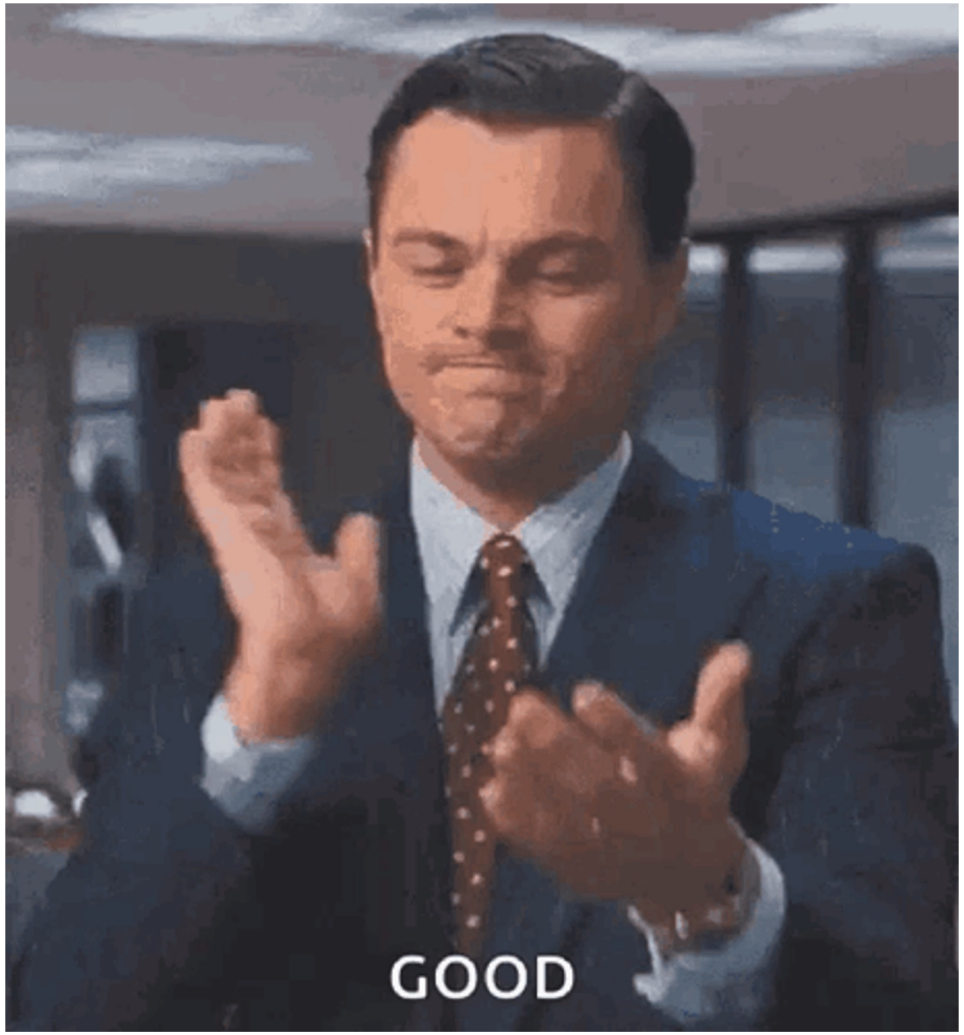
Кодогенерация: добавляем типизированную схему

```yaml
schema: ".codegen/schema.graphql"
documents:
  - "./src/**/*.graphql"
ignoreNoDocuments: true
extensions:
  codegen:
    errorsOnly: true
    generates:
      ./types/schema.ts:
        config:
          defaultScalarType: unknown
          scalars:
            ID: "number"
        plugins:
          - typescript
          - add:
              content: "/* eslint-disable */\n \n /* Auto generated file */ \n "
```

Кодогенерация: смотрим типизированную схему

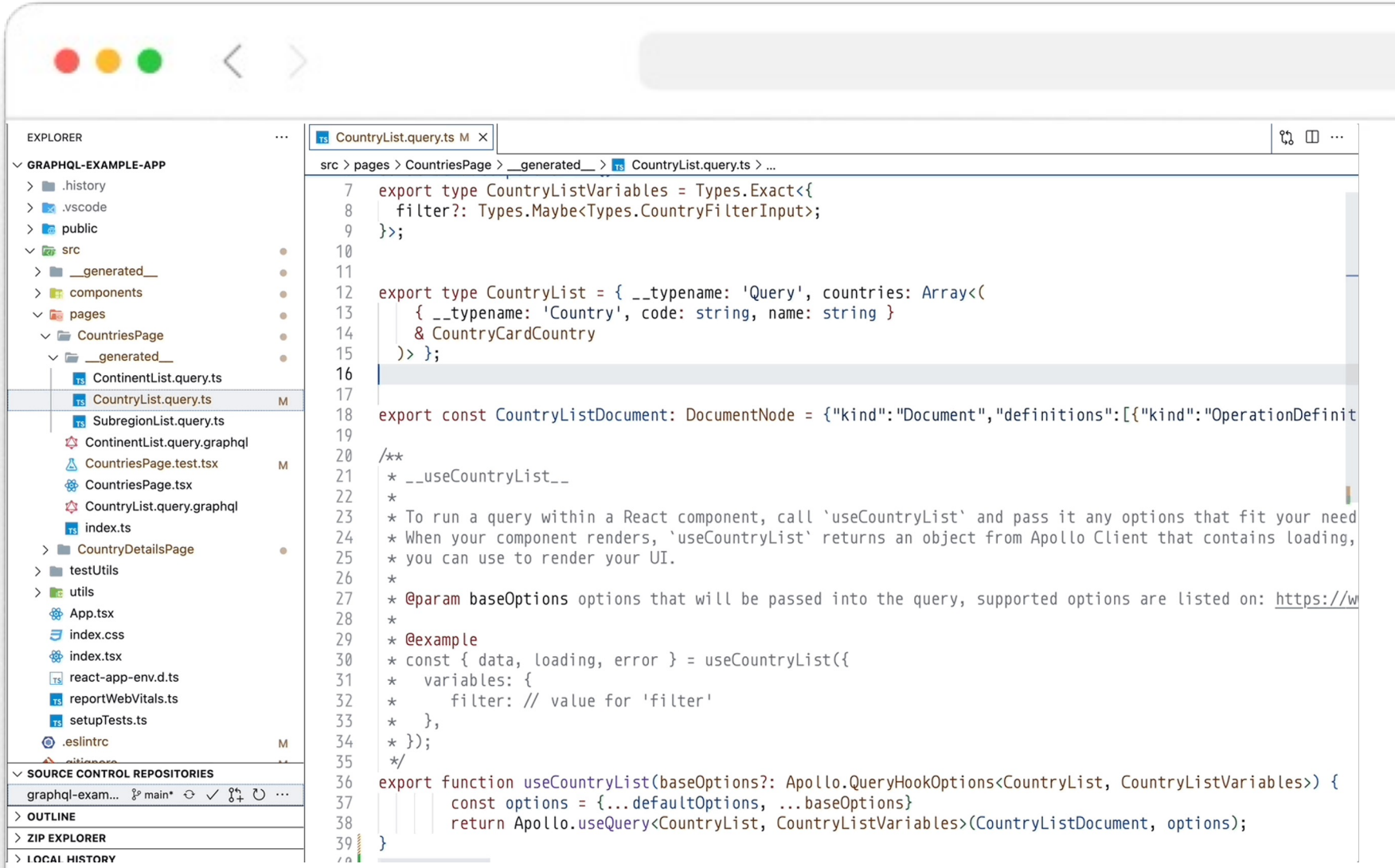



```
{
  "name": "simple-graphql-bootstrap",
  "version": "1.0.0",
  "license": "MIT",
  "type": "module",
  "engines": {
    "node": "≥ v18.18.2"
  },
  ▷ Debug
  "scripts": {
    "schema": "npx ts-node ./scripts/schema/schema.ts",
    "build-plugin": "tsc -p ./scripts/bootstrap-generate/tsconfig.json",
    "build": "tsc -p ./tsconfig.json",
    "test": "jest",
    "lint": "eslint",
    "ts-check": "tsc -p ./tsconfig.json --noEmit",
    "graphql:generate": "graphql-codegen-esm ./.graphqlrc.yml",
    "graphql:generate-watch": "graphql-codegen-esm --watch ./.graphqlrc.yml"
  },
  "dependencies": {
    "@babel/preset-env": "^7.24.3",
    "@babel/preset-typescript": "^7.24.1",
    "graphql": "^16.8.1",
    "graphql-tag": "^2.12.6",
    "prettier": "^3.2.5",
    "typescript": "^5.4.3",
    "undici": "^6.10.1"
  },
  "devDependencies": {
    "@babel/core": "^7.24.3",
    "@graphql-codegen/cli": "^5.0.2",
    "@graphql-codegen/near-operation-file-preset": "^3.0.0",
    "@graphql-codegen/typescript": "^4.0.6",
    "@graphql-eslint/eslint-plugin": "^3.20.1",
```


GOOD

Кодогенерация шаблонного кода

Фича: кодогенерация хуков



Кодогенерация: плагин

```
const plugin: CodegenPlugin<BootStrapRawConfig>['plugin'] = (schema, documents, config) => {
    const documentsNode = documents.reduce((nodes: DocumentNode[], documentFile) => {
        if (documentFile.document) {
            return [documentFile.document, ...nodes]
        }
        return nodes
    }, []);
    const allAst = concatAST(documentsNode);
    const fragments = allAst.definitions.reduce((nodes: FragmentDefinitionNode[], definition) => {
        if (definition.kind === Kind.FRAGMENT_DEFINITION) {
            return [definition, ...nodes]
        }
        return nodes
    }, []);
    const allFragments: LoadedFragment[] = fragments.map(
        (fragment) => ({
            node: fragment,
            name: fragment.name.value,
            onType: fragment.typeCondition.name.value,
            isExternal: false,
        }),
        ...(config.externalFragments || [])
    )
    const visitor = new BootStrapVisitor(schema, allFragments, config, documents);
    const result = oldVisit(allAst, {
        leave: visitor,
        SelectionSet: visitor.addTypenameToDocument(),
    })
    const definitions = result.definitions.filter((t: unknown) => typeof t === 'string').join('\n') as unknown as string;
    const content = [visitor.fragments, definitions].join('\n');

    return {
        prepend: visitor.getImports(),
        content,
    }
}
```

Кодогенерация: пишем плагин

```ts
export const plugin: CodegenPlugin<BootStrapConfig>['plugin'] = (schema, documents, config) => {
    const documentsNode = documents.reduce((nodes: DocumentNode[], documentFile) => {
        if (documentFile.document) {
            return [documentFile.document, ...nodes]
        }
        return nodes
    }, []);
    const allAst = concatAST(documentsNode);
    const fragments = allAst.definitions.reduce((nodes: FragmentDefinitionNode[], definition) => {
        if (definition.kind === Kind.FRAGMENT_DEFINITION) {
            return [definition, ...nodes]
        }
        return nodes
    }, []);
    const allFragments: LoadedFragment[] = fragments.map(
        (fragment) => ({
            node: fragment,
            name: fragment.name.value,
            onType: fragment.typeCondition.name.value,
            isExternal: false,
        }),
        ...(config.externalFragments || [])
    )
    const visitor = new BootStrapVisitor(schema, allFragments, config, documents);
    const result = oldVisit(allAst, {
        // @ts-expect-error: 'leave' is not a NewVisitor['leave']
        leave: visitor,
        SelectionSet: visitor.addTypenameToDocument(),
    })
    const definitions = result.definitions.filter((t: unknown) => typeof t === 'string').join('\n') as unknown as string;
    const content = [visitor.fragments, definitions].join('\n');

    return {
        prepend: visitor.getImports(),
        content: content,
    }
}
```

Кодогенерация: плагин

```
const documentsNode = documents.reduce((nodes: DocumentNode[], documentFile) => {
    if (documentFile.document) {
        return [documentFile.document, ...nodes]
    }
    return nodes
}, []);
const allAst = concatAST(documentsNode);
const fragments = allAst.definitions.reduce((nodes: FragmentDefinitionNode[], definition) => {
    if (definition.kind === Kind.FRAGMENT_DEFINITION) {
        return [definition, ...nodes]
    }
    return nodes
}, []);
const allFragments: LoadedFragment[] = fragments.map(
    (fragment) => ({
        node: fragment,
        name: fragment.name.value,
        onType: fragment.typeCondition.name.value,
        isExternal: false,
    }),
    ...(config.externalFragments || [])
)
const visitor = new BootStrapVisitor(schema, allFragments, config, documents);
const result = oldVisit(allAst, {
    leave: visitor,
    SelectionSet: visitor.addTypenameToDocument(),
})
```

Кодогенерация: плагин

```
const documentsNode = documents.reduce((nodes: DocumentNode[], documentFile) => {
    if (documentFile.document) {
        return [documentFile.document, ...nodes]
    }
    return nodes
}, []);
const allAst = concatAST(documentsNode);
const fragments = allAst.definitions.reduce((nodes: FragmentDefinitionNode[], definition) => {
    if (definition.kind === Kind.FRAGMENT_DEFINITION) {
        return [definition, ...nodes]
    }
    return nodes
}, []);
const allFragments: LoadedFragment[] = fragments.map(
    (fragment) => ({
        node: fragment,
        name: fragment.name.value,
        onType: fragment.typeCondition.name.value,
        isExternal: false,
    }),
    ...(config.externalFragments || [])
)
const visitor = new BootStrapVisitor(schema, allFragments, config, documents);
const result = oldVisit(allAst, {
    leave: visitor,
    SelectionSet: visitor.addTypenameToDocument(),
})
```

Кодогенерация: плагин

```
const documentsNode = documents.reduce((nodes: DocumentNode[], documentFile) => {
    if (documentFile.document) {
        return [documentFile.document, ...nodes]
    }
    return nodes
}, []);
const allAst = concatAST(documentsNode);
const fragments = allAst.definitions.reduce((nodes: FragmentDefinitionNode[], definition) => {
    if (definition.kind === Kind.FRAGMENT_DEFINITION) {
        return [definition, ...nodes]
    }
    return nodes
}, []);
const allFragments: LoadedFragment[] = fragments.map(
    (fragment) => ({
        node: fragment,
        name: fragment.name.value,
        onType: fragment.typeCondition.name.value,
        isExternal: false,
    }),
    ...(config.externalFragments || [])
)
const visitor = new BootStrapVisitor(schema, allFragments, config, documents);
const result = oldVisit(allAst, {
    leave: visitor,
    SelectionSet: visitor.addTypenameToDocument(),
})
```

Кодогенерация: visitor

```
import {
    LoadedFragment,
    ClientSideBaseVisitor,
    ClientSideBasePluginConfig,
} from '@graphql-codegen/visitor-plugin-common'

export class BootStrapVisitor extends ClientSideBaseVisitor<BootStrapRawConfig, BootStrapPluginConfig>
    // code ...
}
```

Кодогенерация: visitor

```
export class BootStrapVisitor extends ClientSideBaseVisitor<BootStrapRawConfig, BootStrapPluginConfig> {
    private fetcherFnName = 'fetcher'
    private templates = [defaultTemplateBuild, defaultTemplateMockBuild, defaultTemplateReactQueryHook]

    protected renderTemplate(props: TemplateProps) {}

    protected buildOperation: BuildOperation = (
        _node,
        _documentVariableName,
        _operation,
        _operationResultType,
        _operationVariablesTypes,
        _hasRequiredVariables
    ) => {
        if (_operation === 'Query' || _operation === 'Mutation') {
            return this.renderTemplate({
                node: _node,
                query: _documentVariableName,
                operationName: _node.name?.value || 'unknown',
                fnName: this.fetcherFnName,
                operation: _operation.toLowerCase(),
                operationResultType: _operationResultType,
                variableTypes: _operationVariablesTypes,
                hasRequiredVariables: _hasRequiredVariables,
            })
        }
        return ''
    }
}
```

Кодогенерация: visitor

```
export class BootStrapVisitor extends ClientSideBaseVisitor<BootStrapRawConfig, BootStrapPluginConfig> {
    private fetcherFnName = 'fetcher'
    private templates = [defaultTemplateBuild, defaultTemplateMockBuild, defaultTemplateReactQueryHook]

    protected renderTemplate(props: TemplateProps) {}

    protected buildOperation: BuildOperation = (
        _node,
        _documentVariableName,
        _operation,
        _operationResultType,
        _operationVariablesTypes,
        _hasRequiredVariables
    ) => {
        if (_operation === 'Query' || _operation === 'Mutation') {
            return this.renderTemplate({
                node: _node,
                query: _documentVariableName,
                operationName: _node.name?.value || 'unknown',
                fnName: this.fetcherFnName,
                operation: _operation.toLowerCase(),
                operationResultType: _operationResultType,
                variableTypes: _operationVariablesTypes,
                hasRequiredVariables: _hasRequiredVariables,
            })
        }
        return ''
    }
}
```

Кодогенерация: visitor

```
export class BootStrapVisitor extends ClientSideBaseVisitor<BootStrapRawConfig, BootStrapPluginConfig> {
    private fetcherFnName = 'fetcher'
    private templates = [defaultTemplateBuild, defaultTemplateMockBuild, defaultTemplateReactQueryHook]

    protected renderTemplate(props: TemplateProps) {}

    protected buildOperation: BuildOperation = (
        _node,
        _documentVariableName,
        _operation,
        _operationResultType,
        _operationVariablesTypes,
        _hasRequiredVariables
    ) => {
        if (_operation === 'Query' || _operation === 'Mutation') {
            return this.renderTemplate({
                node: _node,
                query: _documentVariableName,
                operationName: _node.name?.value || 'unknown',
                fnName: this.fetcherFnName,
                operation: _operation.toLowerCase(),
                operationResultType: _operationResultType,
                variableTypes: _operationVariablesTypes,
                hasRequiredVariables: _hasRequiredVariables,
            })
        }
        return ''
    }
}
```

Кодогенерация: visitor

```
export class BootStrapVisitor extends ClientSideBaseVisitor<BootStrapRawConfig, BootStrapPluginConfig> {
    private fetcherFnName = 'fetcher'
    private templates = [defaultTemplateBuild, defaultTemplateMockBuild, defaultTemplateReactQueryHook]

    protected renderTemplate(props: TemplateProps) {}

    protected buildOperation: BuildOperation = (
        _node,
        _documentVariableName,
        _operation,
        _operationResultType,
        _operationVariablesTypes,
        _hasRequiredVariables
    ) => {
        if (_operation === 'Query' || _operation === 'Mutation') {
            return this.renderTemplate({
                node: _node,
                query: _documentVariableName,
                operationName: _node.name?.value || 'unknown',
                fnName: this.fetcherFnName,
                operation: _operation.toLowerCase(),
                operationResultType: _operationResultType,
                variableTypes: _operationVariablesTypes,
                hasRequiredVariables: _hasRequiredVariables,
            })
        }
        return ''
    }
}
```

Кодогенерация: интерфейсы к Template Builder

```ts
import type { OperationDefinitionNode } from "graphql";

export interface TemplateProps {
    node: OperationDefinitionNode,
    query: string,
    fnName: string;
    /**
     * @value 'query' | 'mutation' | 'subscription'
     */
    operation: string;
    operationResultType: string,
    variableTypes: string,
    hasRequiredVariables: boolean,
}

export interface TemplateFn {
    (props:TemplateProps): string
}
```

Кодогенерация: шаблоны

```
const required = `
    fetch{{operationResultType}} =
        (v: {{variableTypes}}) ⇒
            {{fnName}}.{{operation}}<{{operationResultType}}>
                ({{query}}, v);
`

const partial = `
    fetch{{operationResultType}} =
        (v?: {{variableTypes}}) ⇒
            {{fnName}}.{{operation}}<{{operationResultType}}>
                ({{query}}, v ?? {});
`

const standard = `
    fetch{{operationResultType}} =
        () ⇒
            {{fnName}}.{{operation}}<{{operationResultType}}>({{query}});
`
```

Кодогенерация: шаблоны

```
const t: TemplateFn = (props) => {
    const r = Mustache.render

    if (props.hasRequiredVariables) {
        return r(`export const ${required}`, props)
    }

    if (props.variableTypes && props.node.variableDefinitions?.length) {
        return r(`export const ${partial}`, props)
    }

    return r(`export const ${standard}`, props)
}

export default t
```

Кодогенерация: шаблон fetch функции

```
import type { TemplateFn } from './types.js'
import Mustache from 'mustache'

const required = `
    fetch{{operationResultType}} =
        (v: {{variableTypes}}) ⇒
            {{fnName}}.{{operation}}<{{operationResultType}}>
                ({{query}}, v);
`
const partial = `
    fetch{{operationResultType}} =
        (v?: {{variableTypes}}) ⇒
            {{fnName}}.{{operation}}<{{operationResultType}}>
                ({{query}}, v ?? {});
`
const standard = `
    fetch{{operationResultType}} =
        () ⇒
            {{fnName}}.{{operation}}<{{operationResultType}}>({{query}});
`

const t: TemplateFn = (props) ⇒ {
    const r = Mustache.render
    if (props.hasRequiredVariables) {
        return r(`export const ${required}`, props)
    }
    if (props.variableTypes && props.node.variableDefinitions?.length) {
        return r(`export const ${partial}`, props)
    }
    return r(`export const ${standard}`, props)
}

export default t
```

Кодогенерация: near-operation-file-preset

```
1  yarn add --dev @graphql-codegen/near-operation-file-preset
```

Кодогенерация: настраиваем кодогенерацию файлов

```yaml
schema: ".codegen/schema.graphql"
documents:
  - "./src/**/*.graphql"
ignoreNoDocuments: true
extensions:
  codegen:
    errorsOnly: true
    generates:
      ./types/schema.ts:
        config:
          defaultScalarType: unknown
          scalars:
            ID: "number"
        plugins:
          - typescript
          - add:
              content: "/* eslint-disable */\n \n /* Auto generated file */ \n "
```

Кодогенерация: настраиваем кодогенерацию файлов

```yaml
schema: ".codegen/schema.graphql"
documents:
  - "./src/**/*.graphql"
ignoreNoDocuments: true
extensions:
  codegen:
    errorsOnly: true
    generates:
      ./types/schema.ts:
        config:
          defaultScalarType: unknown
          scalars:
            ID: "number"
        plugins:
          - typescript
          - add:
              content: "/* eslint-disable */\n \n /* Auto generated file */ \n "
      ./src:
        preset: near-operation-file
        presetConfig:
          baseTypesPath: ../types/schema.ts
          folder: _generated_
          extension: .generated.ts
        config:
          skipTypename: true
          nonOptionalTypename: true
          preResolveTypes: false
          namingConvention:
            typeNames: change-case-all#pascalCase
        plugins:
          - typescript-operations
          - './scripts/bootstrap-generate/dist/index.js':
              baseFetch:
                fetcher: './client/api.ts'
```

Кодогенерация: подготовим запросы

first.query.graphql M

src > app > first.query.graphql > characterCard

```graphql
fragment characterCard on Character {
    id
    name
}

Execute Query
query First {
    characters {
        results {
            ...characterCard
        }
    }
}
```

first2.query.graphql M

src > app > first2.query.graphql > First2

```graphql
Execute Query
query First2($page: Int = 1) {
    episodes(page: $page) {
        results {
            name
        }
        info {
            next
        }
    }
}
```

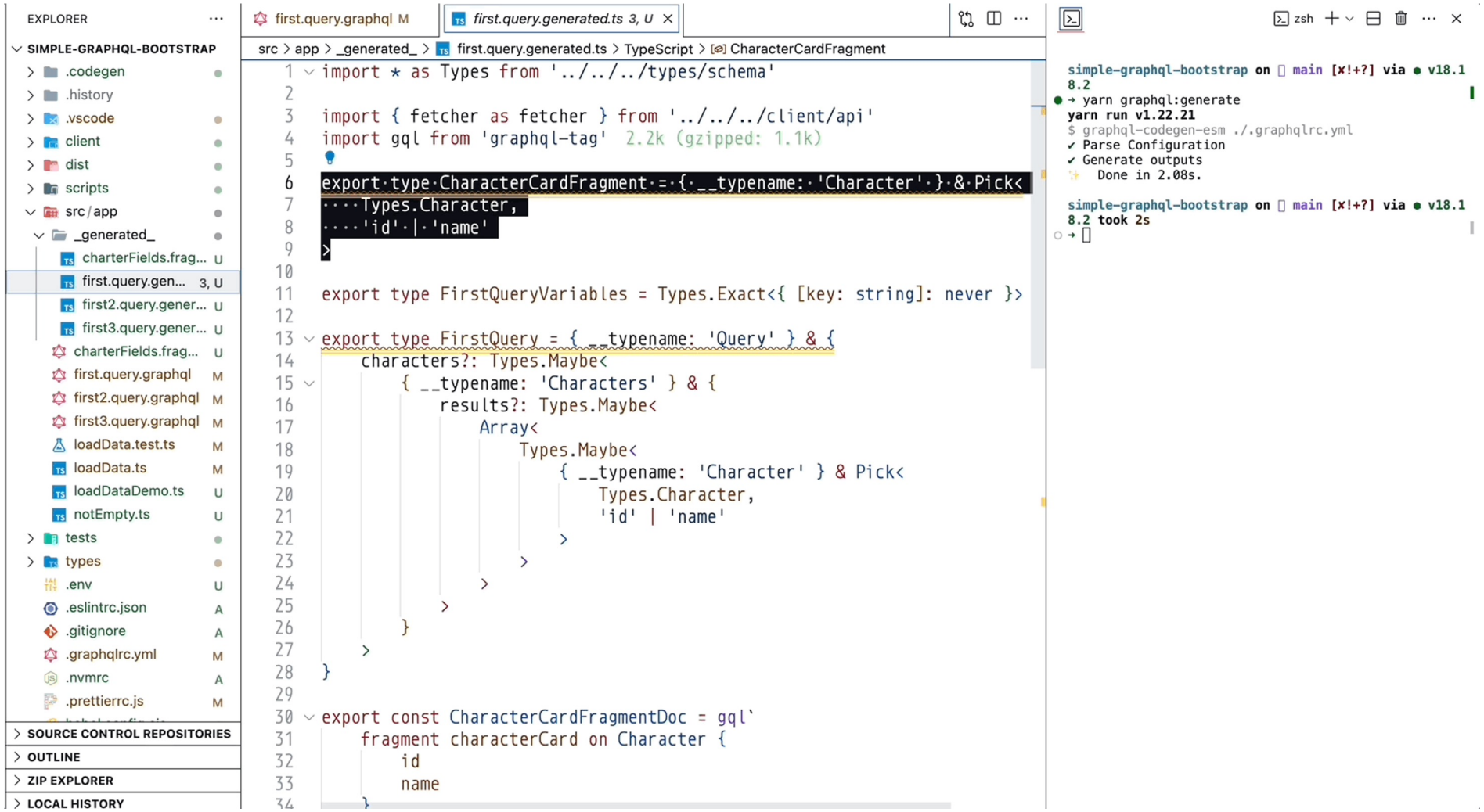
first3.query.graphql M

src > app > first3.query.graphql > First3

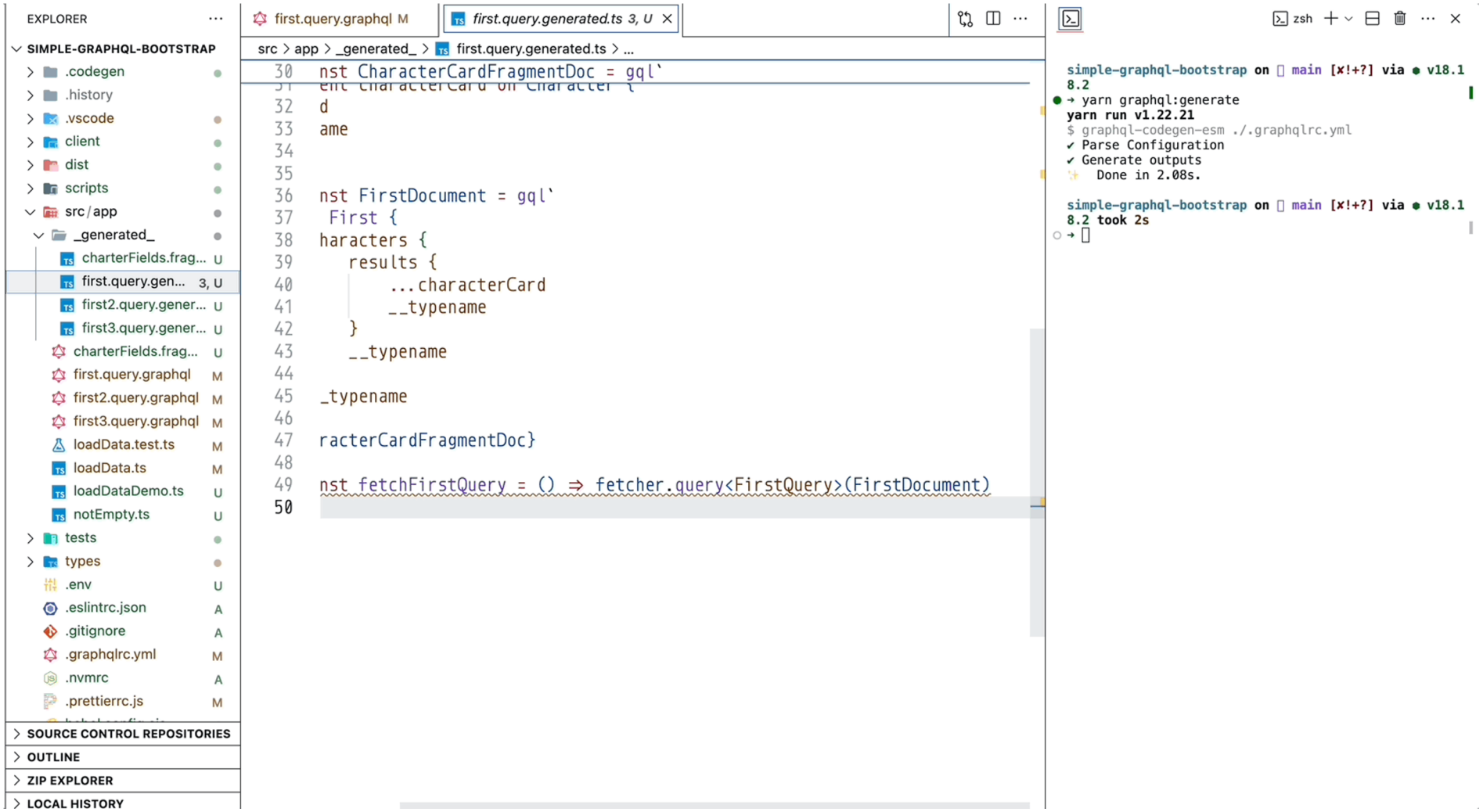
```graphql
Execute Query
query First3($filterName: String!) {
    characters(filter: { name: $filterName }) {
        results {
            name
            status
        }
    }
}
```

Кодогенерация: запускаем генерацию файлов

Кодогенерация: запускаем генерацию файлов

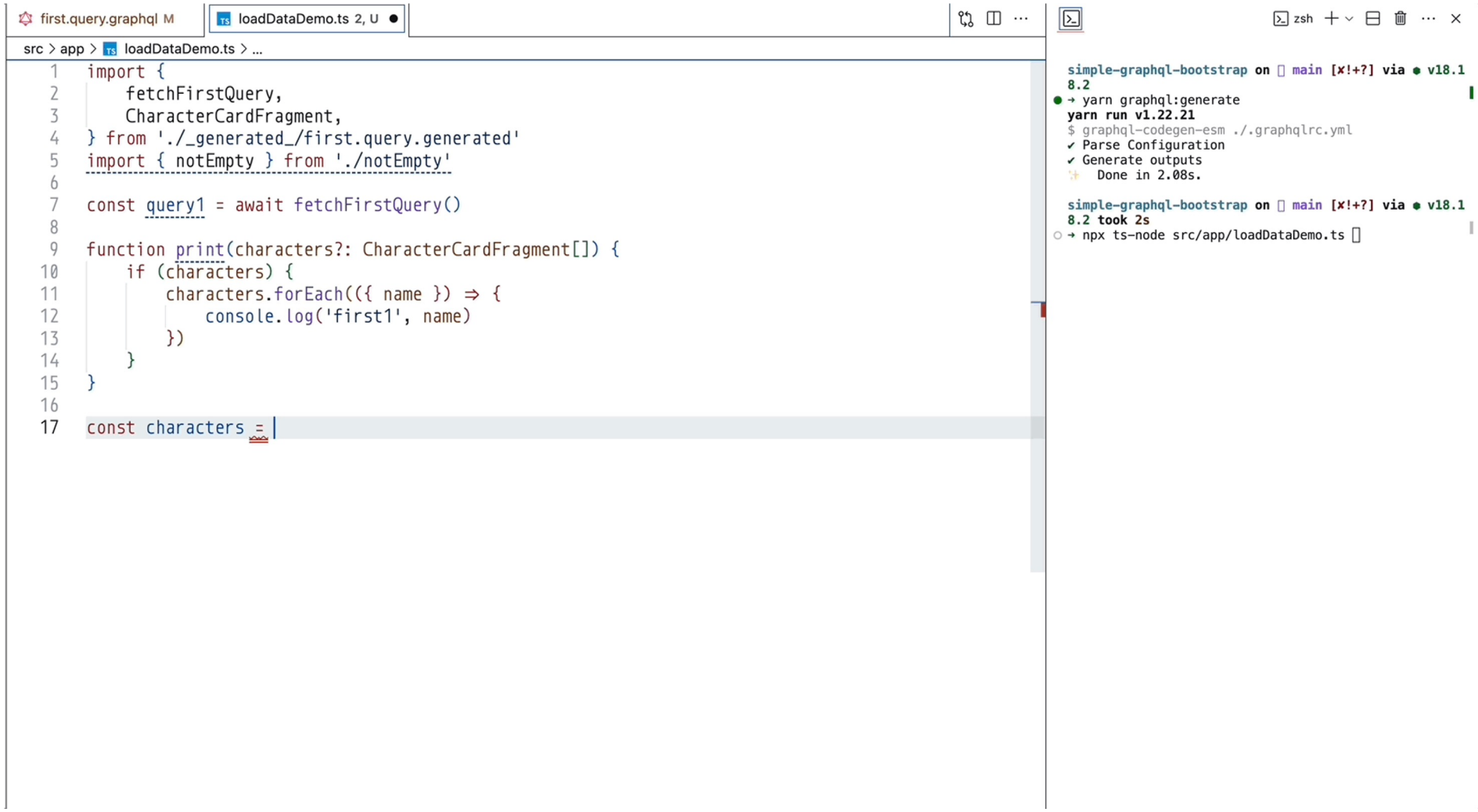


Кодогенерация: запускаем генерацию файлов



Кодогенерация: магия

Кодогенерация: магия



Кодогенерация: __typename ключ к успеху

```
1  const visitor = new BootStrapVisitor(schema, allFragments, config, documents);
2
3  const result = oldVisit(allAst, {
4      // @ts-expect-error: 'leave' is not a NewVisitor['leave']
5      leave: visitor,
```

Кодогенерация: __typename ключ к успеху

```
1  const visitor = new BootStrapVisitor(schema, allFragments, config, documents);
2
3  const result = oldVisit(allAst, {
4      // @ts-expect-error: 'leave' is not a NewVisitor['leave']
5      leave: visitor,
6      SelectionSet: visitor.addTypenameToDocument(),
7  })
```

Кодогенерация: __typename ключ к успеху

```
1  const visitor = new BootStrapVisitor(schema, allFragments, config, documents);
2
3  const result = oldVisit(allAst, {
4      // @ts-expect-error: 'leave' is not a NewVisitor['leave']
5      leave: visitor,
6      SelectionSet: visitor.addTypenameToDocument(),
7  })
```

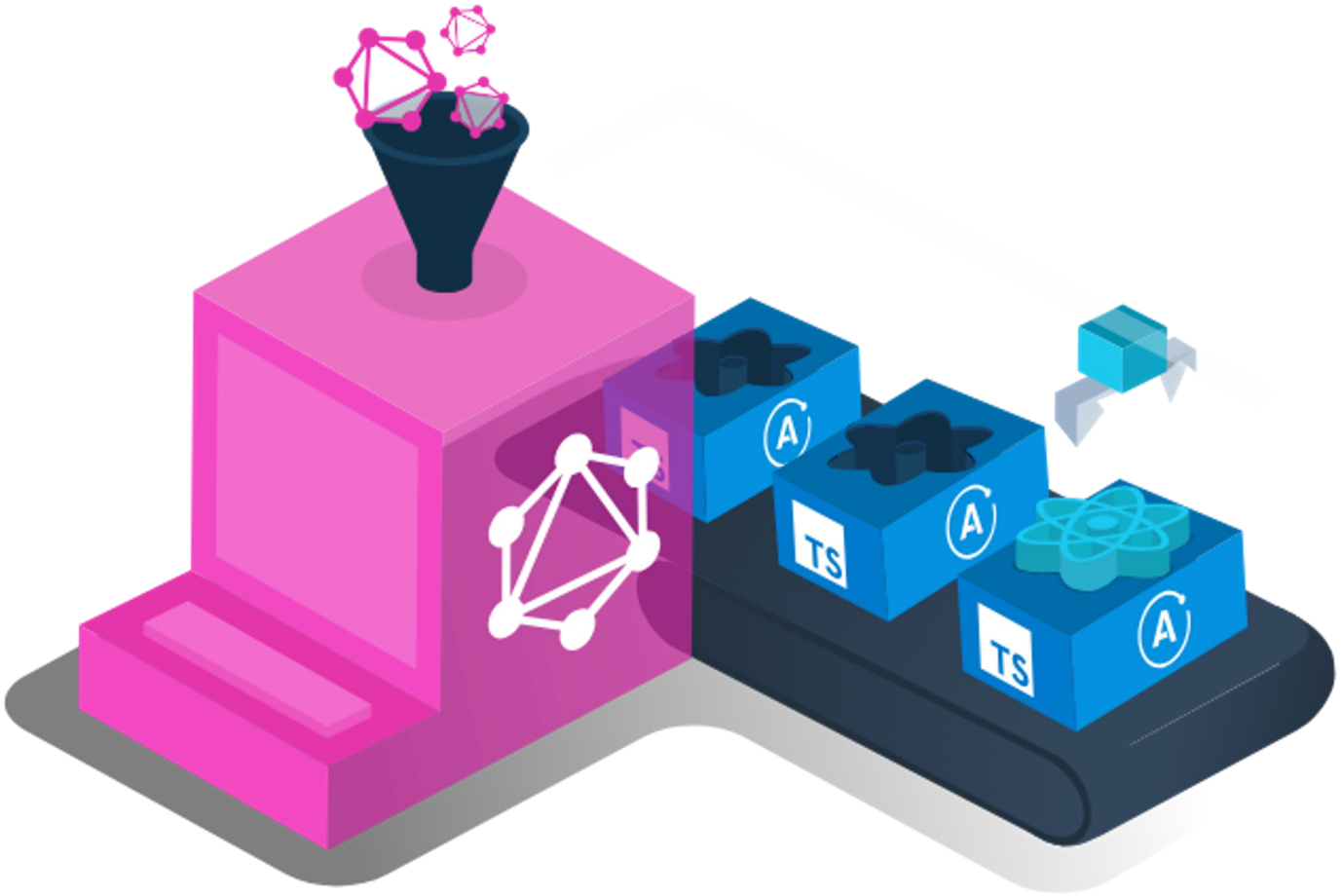
Кодогенерация: __typename ключ к успеху

```
export const CharacterCardFragmentDoc = gql`
    fragment characterCard on Character {
        id
        name
    }
`
export const FirstDocument = gql`
    query First {
        characters {
            results {
                ...characterCard
                __typename
            }
            __typename
        }
        __typename
    }
    ${CharacterCardFragmentDoc}
`
```

Кодогенерация: __typename ключ к успеху

```
90
91  const renderEventItem = (event: SystemNotificationEvent) => {
92    switch (event.__typename) {      You, 8 months ago • HH-187612 systemNotification mode change to gen…
93      case 'CommentCreateBellNotificationItem':
94        return <Views.COMMENT_CREATE event={event} />;
95      case 'MeetingReminderBellNotificationItem':
96        return <Views.MEETING_REMINDER event={event} />;
97      default:
98        return null;
99    }
100 };
```

Наша фабрика по генерации готова



Понадобилось 2 файла и свой шаблон

Тесты

Тесты

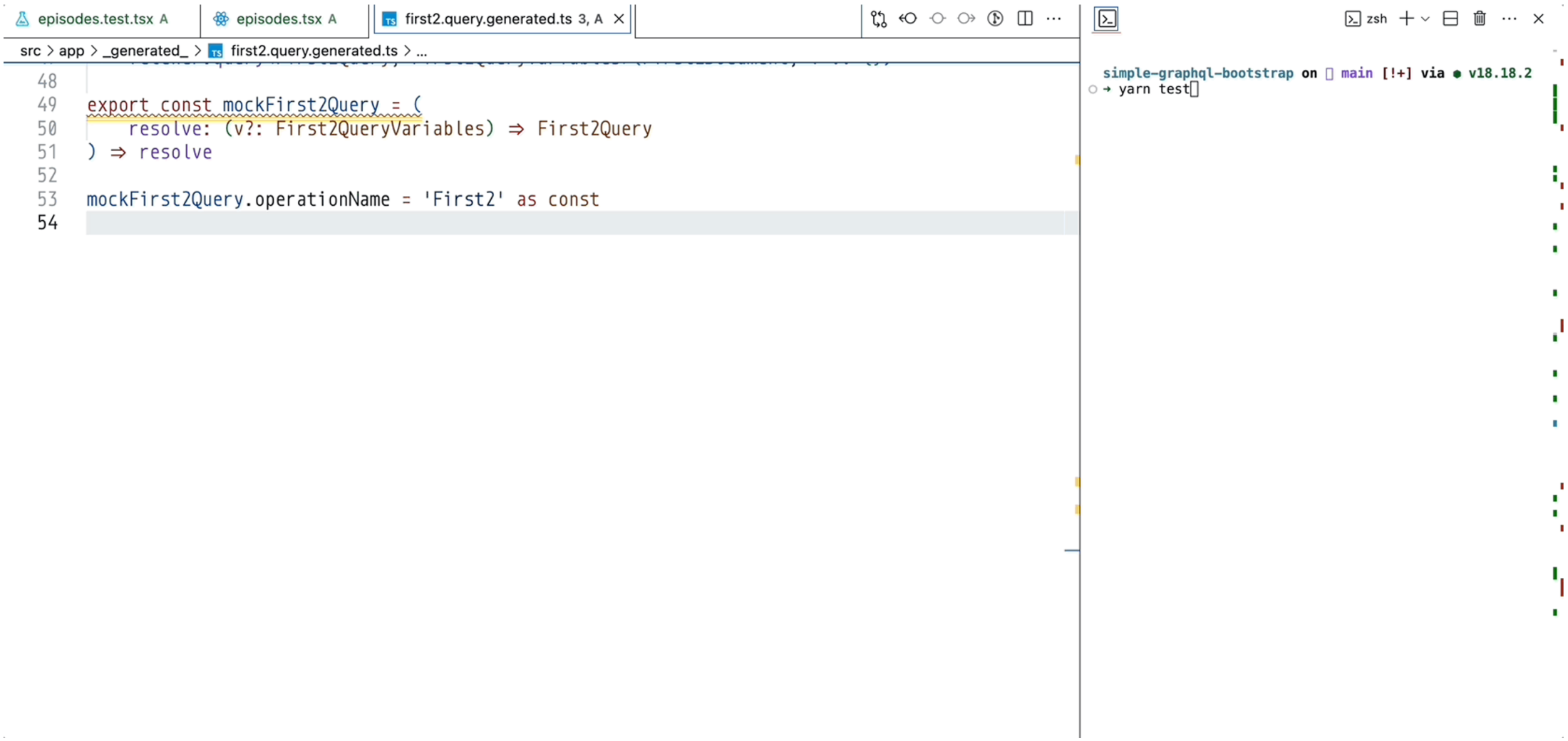
Тесты: генерация мока для операции

```
import type { TemplateFn } from './types.js';
import Mustache from 'mustache';

const r = Mustache.render;

const required = `
    mock{{operationResultType}} =
        (mockFn: (v: {{variableTypes}}) => {{operationResultType}}) => mockFn;
`;

const partial = `
    mock{{operationResultType}} =
        (mockFn: (v?: {{variableTypes}}) => {{operationResultType}}) => mockFn;
`;

const standard = `
    mock{{operationResultType}} =
        (mockFn: () => {{operationResultType}}) => mockFn;
`;


const t: TemplateFn = (props) => {
    if (props.hasRequiredVariables) {
        return r(`export const ${required}`, props);
    }

    if (props.variableTypes && props.node.variableDefinitions?.length) {
        return r(`export const ${partial}`, props);
    }

    return r(`export const ${standard}`, props);
}

export default t;
```

Тесты: генерация мока для операции

```
import type { TemplateFn } from './types.js';
import Mustache from 'mustache';

const r = Mustache.render;

const required = `
    mock{{operationResultType}} =
        (mockFn: (v: {{variableTypes}}) => {{operationResultType}}) => mockFn;
`;

const partial = `
    mock{{operationResultType}} =
        (mockFn: (v?: {{variableTypes}}) => {{operationResultType}}) => mockFn;
`;

const standard = `
    mock{{operationResultType}} =
        (mockFn: () => {{operationResultType}}) => mockFn;
`;


const t: TemplateFn = (props) => {
    if (props.hasRequiredVariables) {
        return r(`export const ${required}`, props);
    }

    if (props.variableTypes && props.node.variableDefinitions?.length) {
        return r(`export const ${partial}`, props);
    }

    return r(`export const ${standard}`, props);
}

export default t;
```

Тесты: поехали



Тесты: мокаем сеть

```
import { Agent, MockAgent, setGlobalDispatcher } from 'undici';
import { fetcher } from '../client/api'

export const createMockRequest = (
    operationName: string,
    disableNetConnect = false
) => {
    const mockAgent = new MockAgent({ connections: 1 })
    const client = mockAgent.get(fetcher.origin)
    const request = client.intercept({
        method: 'POST',
        path: '/graphql',
        query: {
            operationName,
        },
    })
    const clear = async () => {
        setGlobalDispatcher(new Agent())
        await client.close()
    }

    if (disableNetConnect) {
        mockAgent.disableNetConnect()
    }

    setGlobalDispatcher(mockAgent)

    return { request, clear }
}
```

Тесты: мокаем сеть

```
import { Agent, MockAgent, setGlobalDispatcher } from 'undici';
import { fetcher } from '../client/api'

export const createMockRequest = (
    operationName: string,
    disableNetConnect = false
) => {
    const mockAgent = new MockAgent({ connections: 1 })
    const client = mockAgent.get(fetcher.origin)
    const request = client.intercept({
        method: 'POST',
        path: '/graphql',
        query: {
            operationName,
        },
    })
    const clear = async () => {
        setGlobalDispatcher(new Agent())
        await client.close()
    }

    if (disableNetConnect) {
        mockAgent.disableNetConnect()
    }

    setGlobalDispatcher(mockAgent)

    return { request, clear }
}
```

Тесты: мокаем сеть

```
import { Agent, MockAgent, setGlobalDispatcher } from 'undici';
import { fetcher } from '../client/api'

export const createMockRequest = (
    operationName: string,
    disableNetConnect = false
) => {
    const mockAgent = new MockAgent({ connections: 1 })
    const client = mockAgent.get(fetcher.origin)
    const request = client.intercept({
        method: 'POST',
        path: '/graphql',
        query: {
            operationName,
        },
    })
    const clear = async () => {
        setGlobalDispatcher(new Agent())
        await client.close()
    }

    if (disableNetConnect) {
        mockAgent.disableNetConnect()
    }

    setGlobalDispatcher(mockAgent)

    return { request, clear }
}
```

Тесты: мок GraphQL операций

```ts
import { createMockRequest } from './createMockRequest'

const mockGraphQL = (operationName: string, disableNetConnect = true) => {
    const { request, clear } = createMockRequest(
        operationName,
        disableNetConnect
    )

    return {
        reply<T extends (...args: any[]) => object>(
            resolver: ResolveFn<T>
        ) {
            request.reply(function replyHandler(props) {
                const body = typeof props.body === 'string'
                ? JSON.parse(props.body)
                    : undefined

                return {
                    statusCode: 200,
                    data: { data: resolver(body?.variables ?? undefined) },
                }
            })
        },
        replyWithError: <S extends number, E extends object>(
            status: S,
            error: E
        ) => request.reply(status, error),
        clear,
    }
}
```

Тесты: мок GraphQL операций

```ts
import { createMockRequest } from './createMockRequest'

const mockGraphQL = (operationName: string, disableNetConnect = true) => {
    const { request, clear } = createMockRequest(
        operationName,
        disableNetConnect
    )

    return {
        reply<T extends (...args: any[]) => object>(
            resolver: ResolveFn<T>
        ) {
            request.reply(function replyHandler(props) {
                const body = typeof props.body === 'string'
                ? JSON.parse(props.body)
                    : undefined

                return {
                    statusCode: 200,
                    data: { data: resolver(body?.variables ?? undefined) },
                }
            })
        },
        replyWithError: <S extends number, E extends object>(
            status: S,
            error: E
        ) => request.reply(status, error),
        clear,
    }
}
```

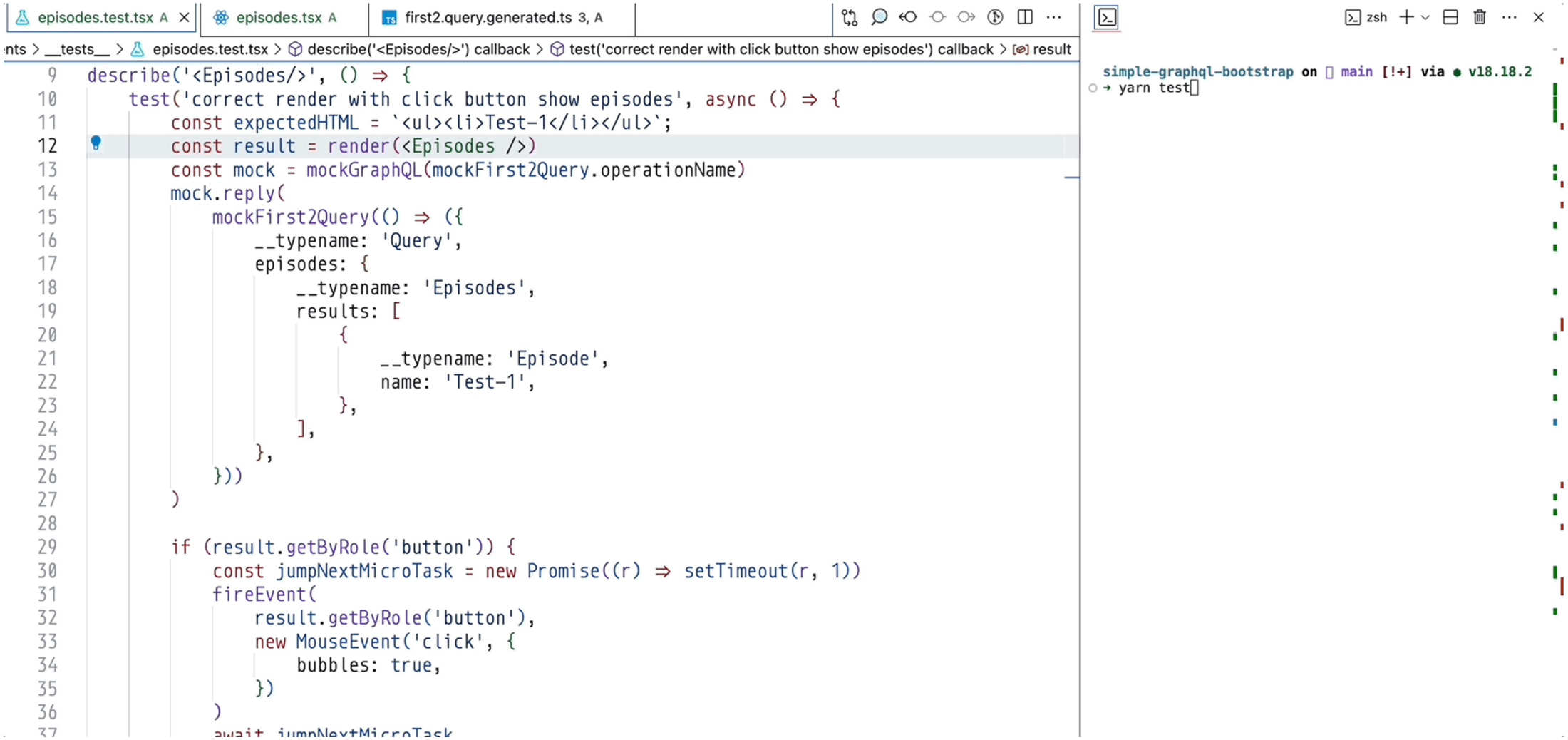
Тесты: компонент

```tsx
import { fetchFirst2Query } from '../_generated_/first2.query.generated'
import { EpisodeListFragment } from '../_generated_/episodesList.fragment.generated'
import { notEmpty } from '../notEmpty'

const Episodes = () => {
    const [episodes, setEpisodes] = React.useState<EpisodeListFragment[]>([])
    const handleClickLoadButton = async () => {
        const { data } = await fetchFirst2Query({ page: 1 })
        const episodes = data?.episodes?.results?.filter(notEmpty)
        if (episodes) {
            setEpisodes(episodes)
        }
    }

    return (
        <ul>
            {episodes.map((episode) => (
                <li key={episode.name}>{episode.name}</li>
            ))}
            {!episodes.length && (
                <button role="button" onClick={handleClickLoadButton}>
                    show episodes
                </button>
            )}
        </ul>
    )
}
```

Тесты: компонент

```
1  import { fetchFirst2Query } from '../_generated_/first2.query.generated'
2  import { EpisodeListFragment } from '../_generated_/episodesList.fragment.generated'
3  import { notEmpty } from '../notEmpty'
4
5  const Episodes = () => {
6      const [episodes, setEpisodes] = React.useState<EpisodeListFragment[]>([])
7      const handleClickLoadButton = async () => {
8          const { data } = await fetchFirst2Query({ page: 1 })
9          const episodes = data?.episodes?.results?.filter(notEmpty)
10         if (episodes) {
11             setEpisodes(episodes)
12         }
13     }
14
15     return (
16         <ul>
17             {episodes.map((episode) => (
18                 <li key={episode.name}>{episode.name}</li>
19             ))}
20             {!episodes.length && (
21                 <button role="button" onClick={handleClickLoadButton}>
22                     show episodes
23                 </button>
24             )}
25         </ul>
26     )
27 }
```

Тесты: запуск



Итог

Итог

01.

Клиент

Итог

01.

Клиент

02.

Плейграунд

Итог

01.

Клиент

02.

Плейграунд

03.

Кодогенерация

Итог

01.

Клиент

02.

Плейграунд

03.

Кодогенерация

04.

Тесты

Все готово.
Что дальше?

Фиксируем договоренности

Фиксируем договоренности

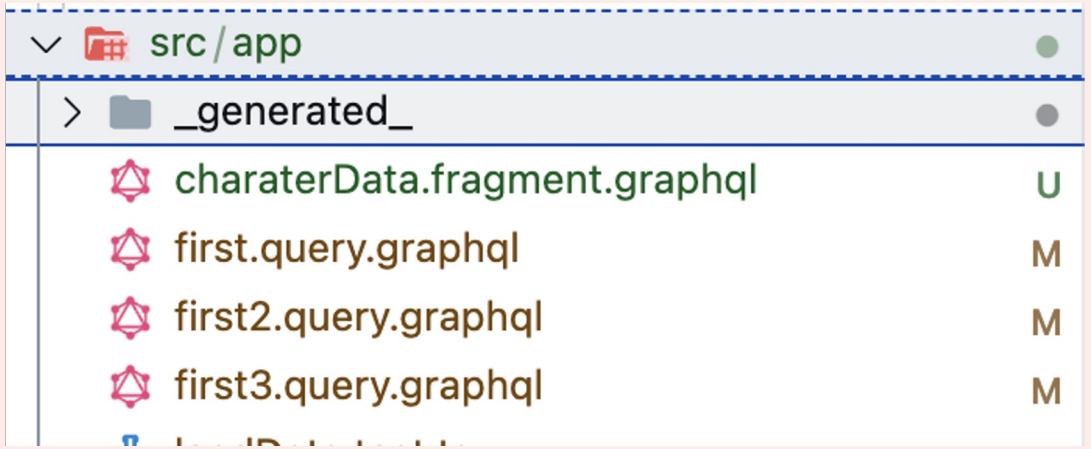
+

Операции и фрагменты только в файлах

Фиксируем договоренности

+

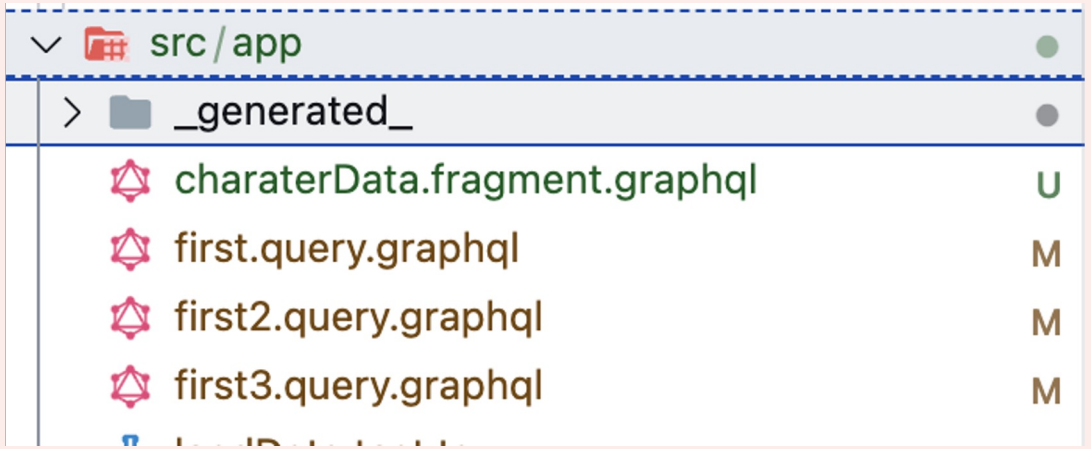
Операции и фрагменты только в файлах



Фиксируем договоренности

+

Операции и фрагменты только в файлах



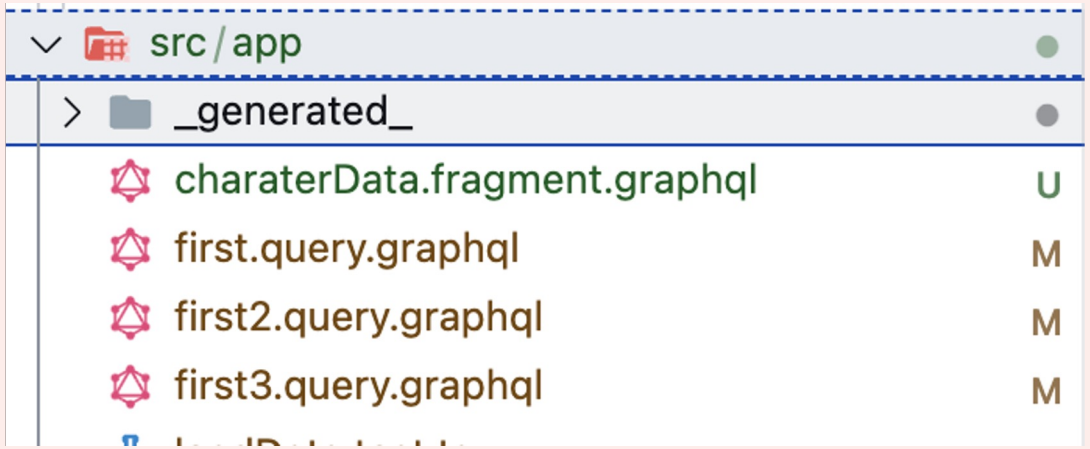
–

Не пишем запросы во вью слое, не собираем template literals запросы

Фиксируем договоренности

+

Операции и фрагменты только в файлах



–

Не пишем запросы во вью слое, не собираем template literals запросы

```
function App() {
  const allFilmsWithVariablesQueryDocument = graphql(/* GraphQL */ `
    query allFilmsWithVariablesQuery($first: Int!) {
      allFilms(first: $first) {
        edges {
          node {
            id
            title
          }
        }
      }
    }
`)
  const { data } = useQuery({
    queryKey: ['films'],
    queryFn: async () =>
      request(
        'https://swapi-graphql.netlify.app/.netlify/functions/index',
        allFilmsWithVariablesQueryDocument,
        { first: 10 },
      ),
  })
  // ...
}
```

Eslint

```
1  yarn add --dev @graphql-eslint/eslint-plugin
```

Rules

```
"prettier/prettier": [2, { "parser": "graphql" }],
"spaced-comment": "off",
"@graphql-eslint/avoid-duplicate-fields": "error",
"@graphql-eslint/executable-definitions": "error",
"@graphql-eslint/fields-on-correct-type": "error",
"@graphql-eslint/fragments-on-composite-type": "error",
"@graphql-eslint/known-argument-names": "error",
"@graphql-eslint/known-directives": "error",
"@graphql-eslint/known-type-names": "error",
"@graphql-eslint/no-anonymous-operations": "error",
"@graphql-eslint/no-deprecated": "error",
"@graphql-eslint/no-unused-variables": "error",
"@graphql-eslint/provided-required-arguments": "error",
"@graphql-eslint/scalar-leafs": "error",
"@graphql-eslint/unique-argument-names": "error",
"@graphql-eslint/unique-input-field-names": "error",
"@graphql-eslint/unique-variable-names": "error",
"@graphql-eslint/value-literals-of-correct-type": "error",
"@graphql-eslint/variables-are-input-types": "error",
"@graphql-eslint/variables-in-allowed-position": "error",
"@graphql-eslint/require-id-when-available": "off",
```

Rules

```
"prettier/prettier": [2, { "parser": "graphql" }],
"spaced-comment": "off",
"@graphql-eslint/avoid-duplicate-fields": "error",
"@graphql-eslint/executable-definitions": "error",
"@graphql-eslint/fields-on-correct-type": "error",
"@graphql-eslint/fragments-on-composite-type": "error",
"@graphql-eslint/known-argument-names": "error",
"@graphql-eslint/known-directives": "error",
"@graphql-eslint/known-type-names": "error",
"@graphql-eslint/no-anonymous-operations": "error",
"@graphql-eslint/no-deprecated": "error",
"@graphql-eslint/no-unused-variables": "error",
"@graphql-eslint/provided-required-arguments": "error",
"@graphql-eslint/scalar-leafs": "error",
"@graphql-eslint/unique-argument-names": "error",
"@graphql-eslint/unique-input-field-names": "error",
"@graphql-eslint/unique-variable-names": "error",
"@graphql-eslint/value-literals-of-correct-type": "error",
"@graphql-eslint/variables-are-input-types": "error",
"@graphql-eslint/variables-in-allowed-position": "error",
"@graphql-eslint/require-id-when-available": "off",
```

Rules

```
// 20 is backend constant value
"@graphql-eslint/selection-set-depth": ["error", { "maxDepth": 20 }],
"@graphql-eslint/naming-convention": [
  "error",
  {
    "VariableDefinition": "camelCase",
    "OperationDefinition": {
      "style": "PascalCase",
      "forbiddenPrefixes": ["Query", "Mutation", "Subscription", "Get"],
      "forbiddenSuffixes": ["Query", "Mutation", "Subscription"]
    },
    "FragmentDefinition": {
      "style": "camelCase",
      "forbiddenPrefixes": ["Fragment"],
      "forbiddenSuffixes": ["Fragment"]
    }
  }
]
```

Rules

```
{
  "@graphql-eslint/match-document-filename": [
    "error",
    {
      "fileExtension": ".graphql",
      "query": { "style": "camelCase", "suffix": ".query" },
      "mutation": { "style": "camelCase", "suffix": ".mutation" },
      "fragment": { "style": "camelCase", "suffix": ".fragment" }
    }
  ]
}
```

Rules

```
{
  "@graphql-eslint/match-document-filename": [
    "error",
    {
      "fileExtension": ".graphql",
      "query": { "style": "camelCase", "suffix": ".query" },
      "mutation": { "style": "camelCase", "suffix": ".mutation" },
      "fragment": { "style": "camelCase", "suffix": ".fragment" }
    }
  ]
}
```

Реализую задуманное

Важно!

Работаем только на уровне сети

Важно!

Не трогаем слой view

Реализация

1

Пишу Query

Реализация

1

Пишу Query

2

Подменяю клиент

Реализация

1

Пишу Query

2

Подменяю клиент

3

Маппинг данных для стора

Подключаем наш axios

```
import axios from 'axios'
import { type ASTNode, type OperationDefinitionNode } from 'graphql'
import type { RequestFn } from './types'

const isOperation = (node: ASTNode): node is OperationDefinitionNode => node.kind === 'OperationDefinition'

const getOperationName = (node: ASTNode): string => isOperation(node) ? node.name!.value : 'unknown'

export const request:<U extends URL>(url: U) => RequestFn = (url) => async (query, variables) => {
        const isDocumentNode = typeof query !== 'string';
        const operationName = isDocumentNode ? getOperationName(query.definitions[0]) : 'unknown';
        operationName && url.searchParams.append('operationName', operationName)
+       const { data } = await axios.request({
+           url: `${url}`,
+           method: 'post',
+           headers: {
+               'Content-Type': 'application/json',
+           },
+           data: {
+               query: isDocumentNode? query.loc?.source.body : query,
+               variables: variables ?? null,
+               operationName,
+           }
+       })
        return data
    }
```

Старый fetcher

personsMap.js

Users > i.gorskiy > projects > tms > tms-static > app > actions > ats > personsMap > personsMap.js

```
108  addReaction(
172  );
173
174  addReaction(
175    PERSONS_MAP_LOAD,
176    async (
177      { getState, dispatch, fetcher },
178      { vacancyId, statusId, offset, query, resetPersonData, mergePersonId, fromVacancy }
179    ) => {
180      const isMergeRequest = mergePersonId !== null;
181
182 >    if ((offset === 0 || isMergeRequest) && resetPersonData !== false) {…
184      }
185
186 >    if (resetPersonData) {…
188      }
189
190      dispatch(fetchPersons());
191
192      try {
193 >      const { data } = await fetcher.get(getUrl({ vacancyId, statusId, mergePersonId }), {      prizemlenie, 6 years ago • HH-81915 create mer
196        });
197
198        const actions = [];
199
200 >      if (offset === 0 && resetPersonData !== false) {…
204        }
205
```

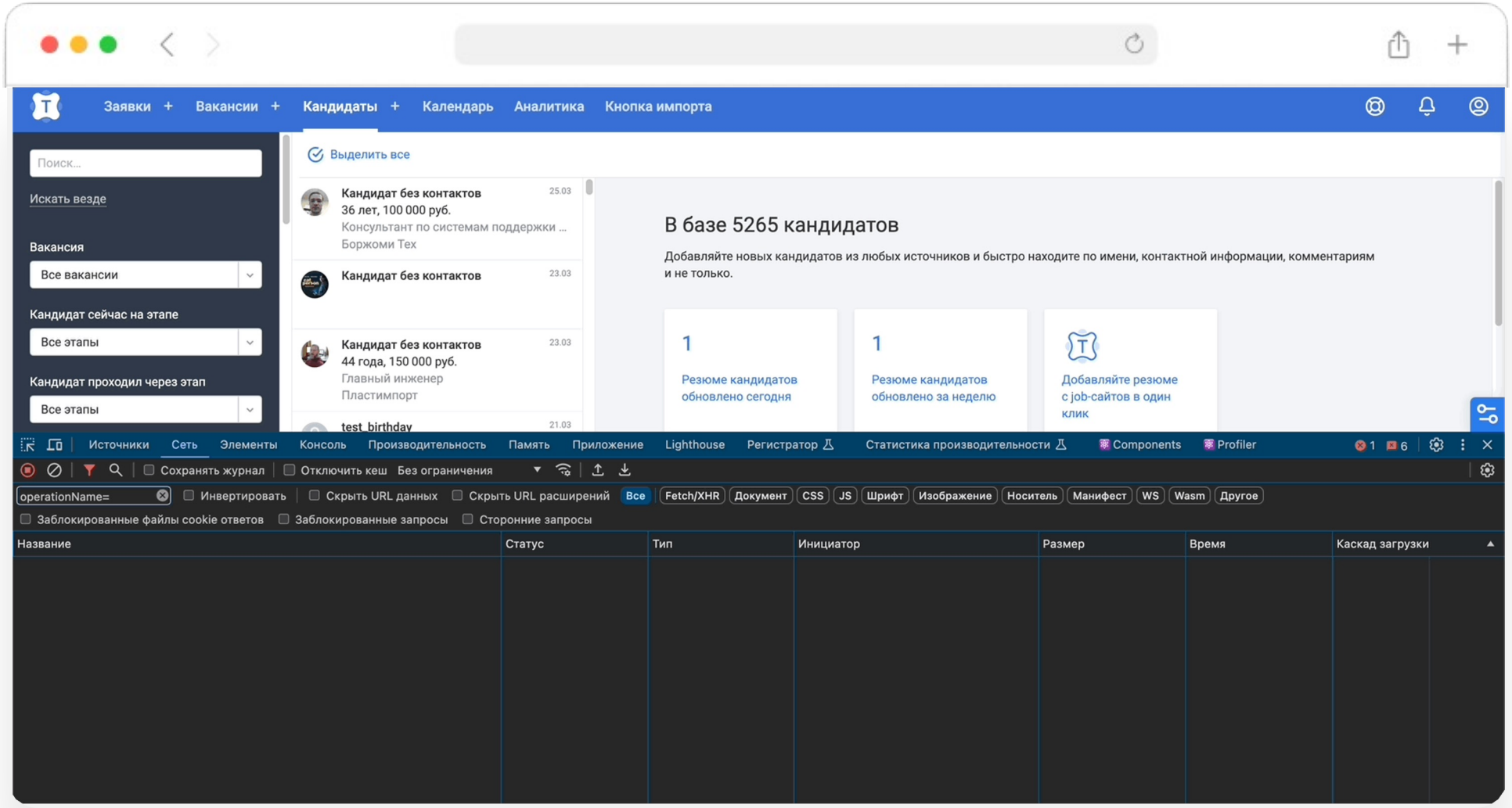
Подменяем новый fetcher

personsMap.js

Users > i.gorskiy > projects > tms > tms-static > app > actions > ats > personsMap > personsMap.js

```
108  addReaction(
110    'PERSONS',
111    async ({ getState, dispatch, fetcher }, { resetPersonData, query, nextPersons }) => {
112      const resetActions = [resetPersons(), resetPersonsMap()];
113      const actions = [];
114      dispatch(fetchPersons());
115
116  >   if (!nextPersons && resetPersonData) {…
118      }
119
120      try {
121        const { endCursor, hasNextPage } = getState().ats.persons.data?.pageInfo || {};
122        const variables = { filter: buildFilterInput(query) };
123
124  >     if (nextPersons && hasNextPage && !resetPersonData) {…
127        }
128
129        const { data } = await fetcher.query(PersonListQuery, variables);
130        const { persons: personsData } = data;
131        const { items, filters, ...persons } = personsData;
132
133  >     if (data?.vacancies?.hasActive) {…
139        }
140
141  >     if (data?.formValidation) {…
143        }
144
145  >     if (data?.experiments) {…
147        }
```

Готовая страница на GraphQL



Profit!

Выводы

Выводы

Фронт потратил **в 2 раза меньше** времени на разработку

Выводы

Не влипли в молодежные фреймворки

Выводы

Получили экспертизу
в технологии GraphQL

Выводы

Вместе разобрались в магии по шагам

Выводы

Шаги легко повторить

Выводы

Шаги легко повторить

Спасибо за внимание!