



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

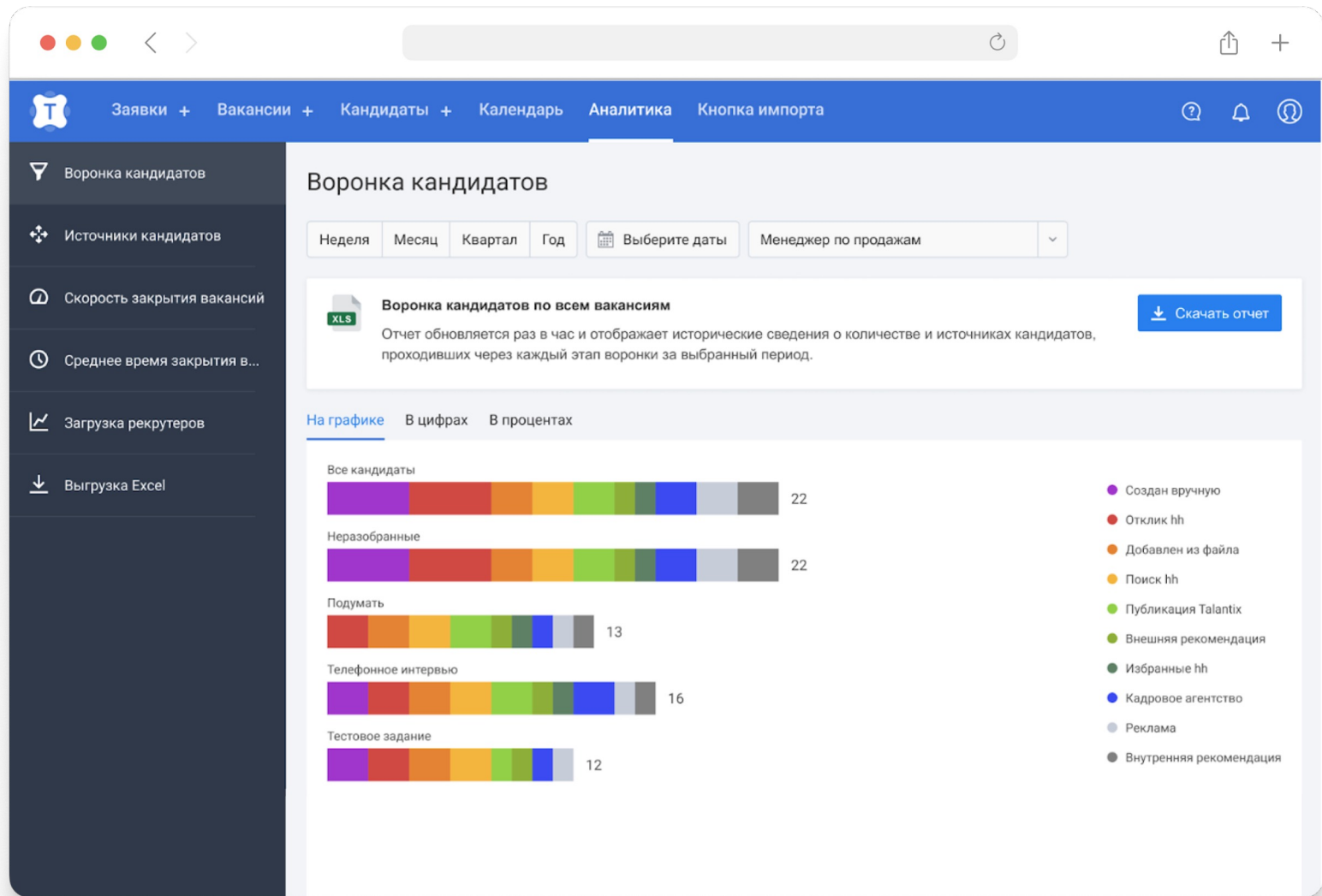
Ведущий разработчик hh.ru

- 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

REST API



GET /astronaut

GET /rocket

GET /moon

GraphQL Gateway



`/graphql`



underFETCHING

UNDERFECTHING

OVERfetching

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

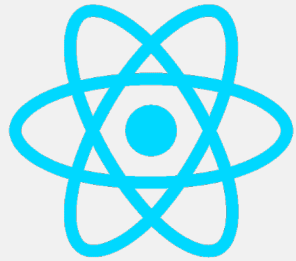
RESOLVE

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

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

О проекте

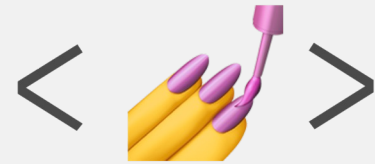
Frontend



React



Redux



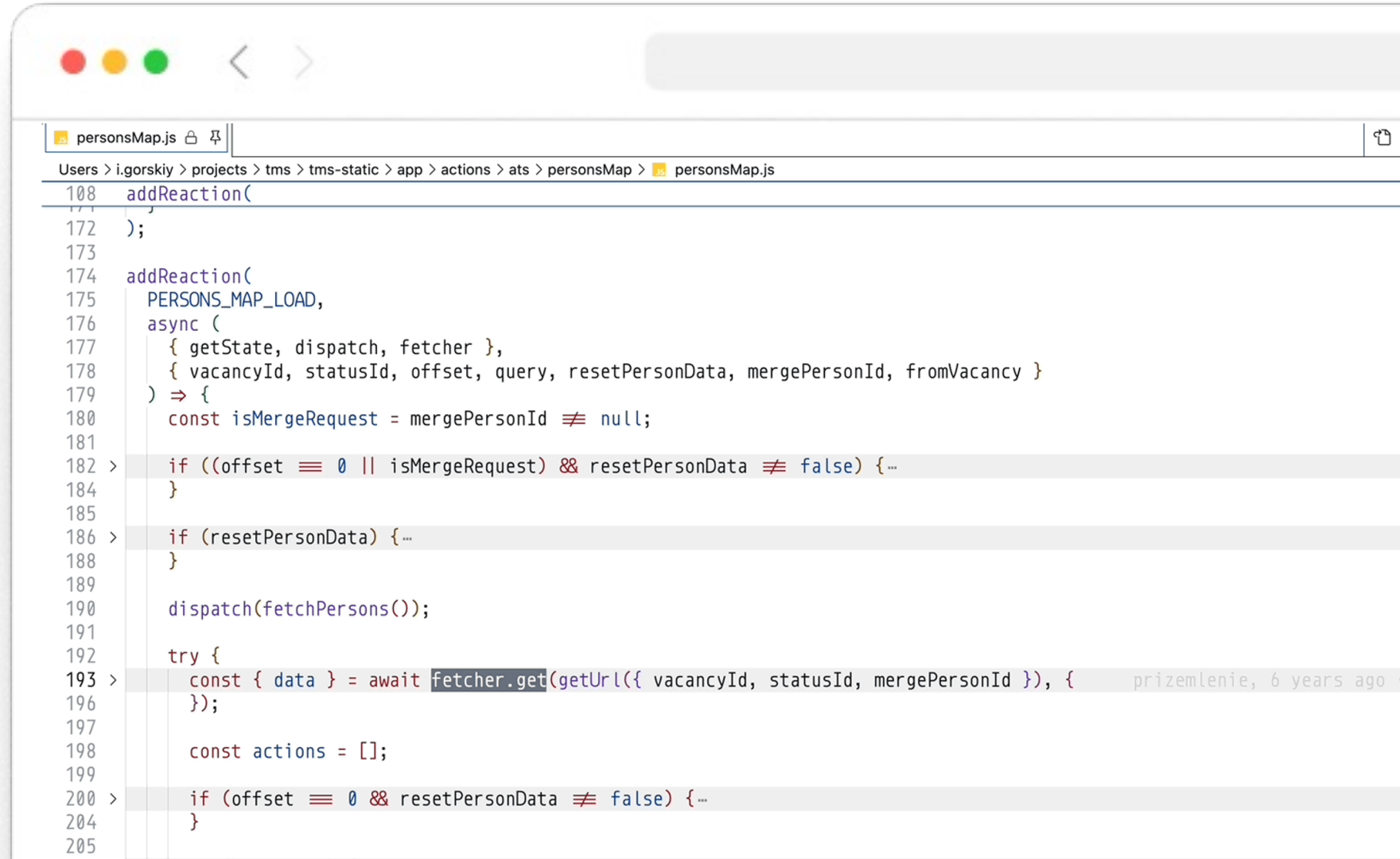
styled
components

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(
171  );
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

```
1 import { errorRecord } from 'Models/records';
2
3 import t from 'Translations/translations';
4
5 const refreshInterceptor =
6   ({ store }) =>
7   (error) => {
8     if (error?.response?.status === 428) {
9       if (error.config.method === 'get') {
10         window.location.reload(true);
11         return error;
12       }
13
14       store.dispatch(errorRecord(t('update.new.version'), true));
15       return error;
16     }
17
18     return Promise.reject(error);
19   };
20
21 export default {
22   inject: { response: ['store'] },
23   request: [],
24   response: [null, refreshInterceptor],
25 };
```

Interceptors

```
1 import { start, stop } from 'Models/progress';
2
3 import { statusResolver } from 'Utils/fetcher/interceptors/helper';
4
5 const requestStart = (status) => (config) => {
6   start();
7   return statusResolver[status](config);
8 };
9
10 const responseStop = (status) => (response) => {
11   stop();
12   return statusResolver[status](response);
13 };
14
15 export default {
16   request: [requestStart('fulfilled'), responseStop('rejected')],
17   response: [responseStop('fulfilled'), responseStop('rejected')],
18 };
```

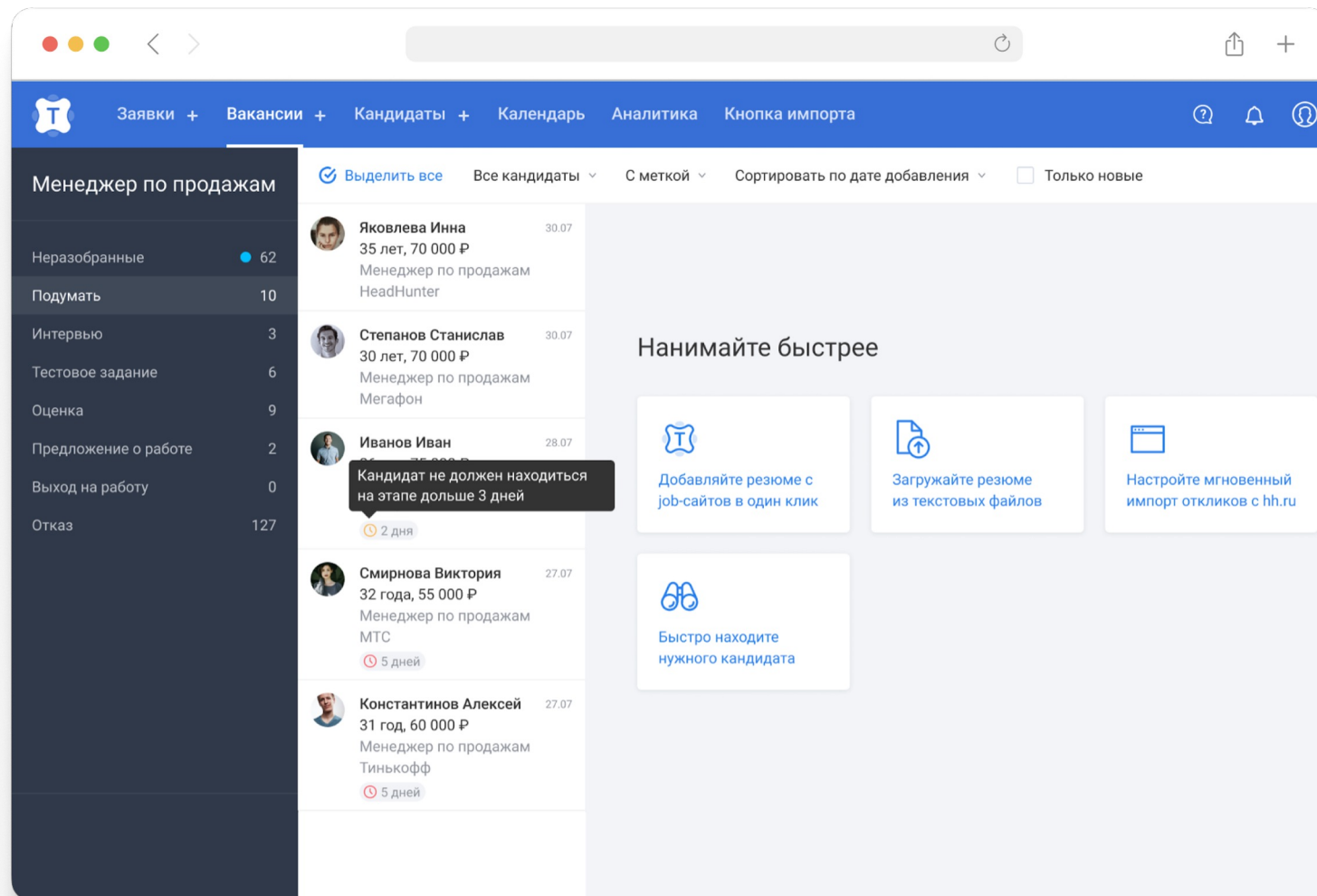
Постановка проблемы

Проблемы текущего подхода

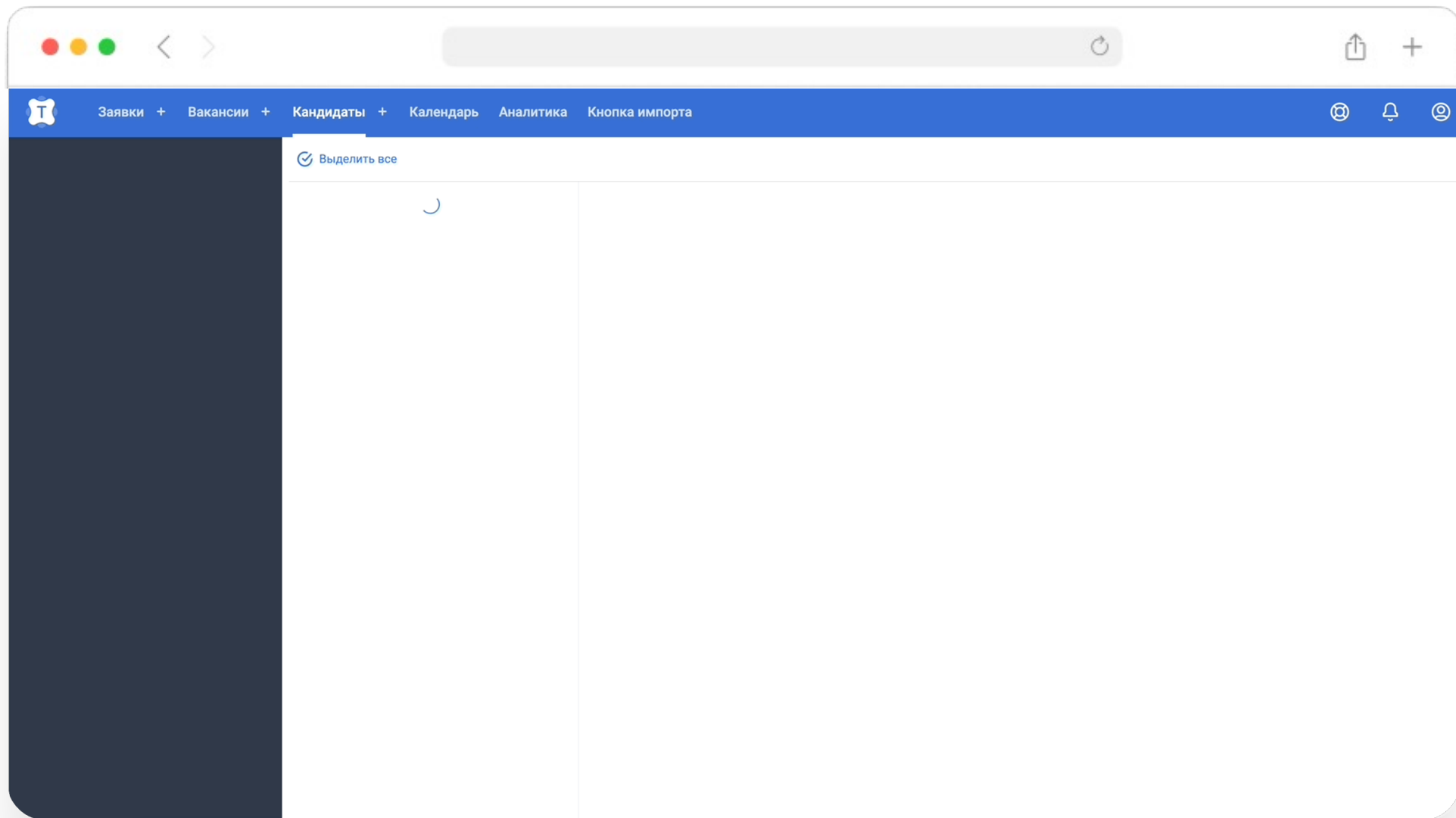
1

AJAX страницы

Одним запросом получаем все данные для страницы



Список кандидатов



Проблемы текущего подхода

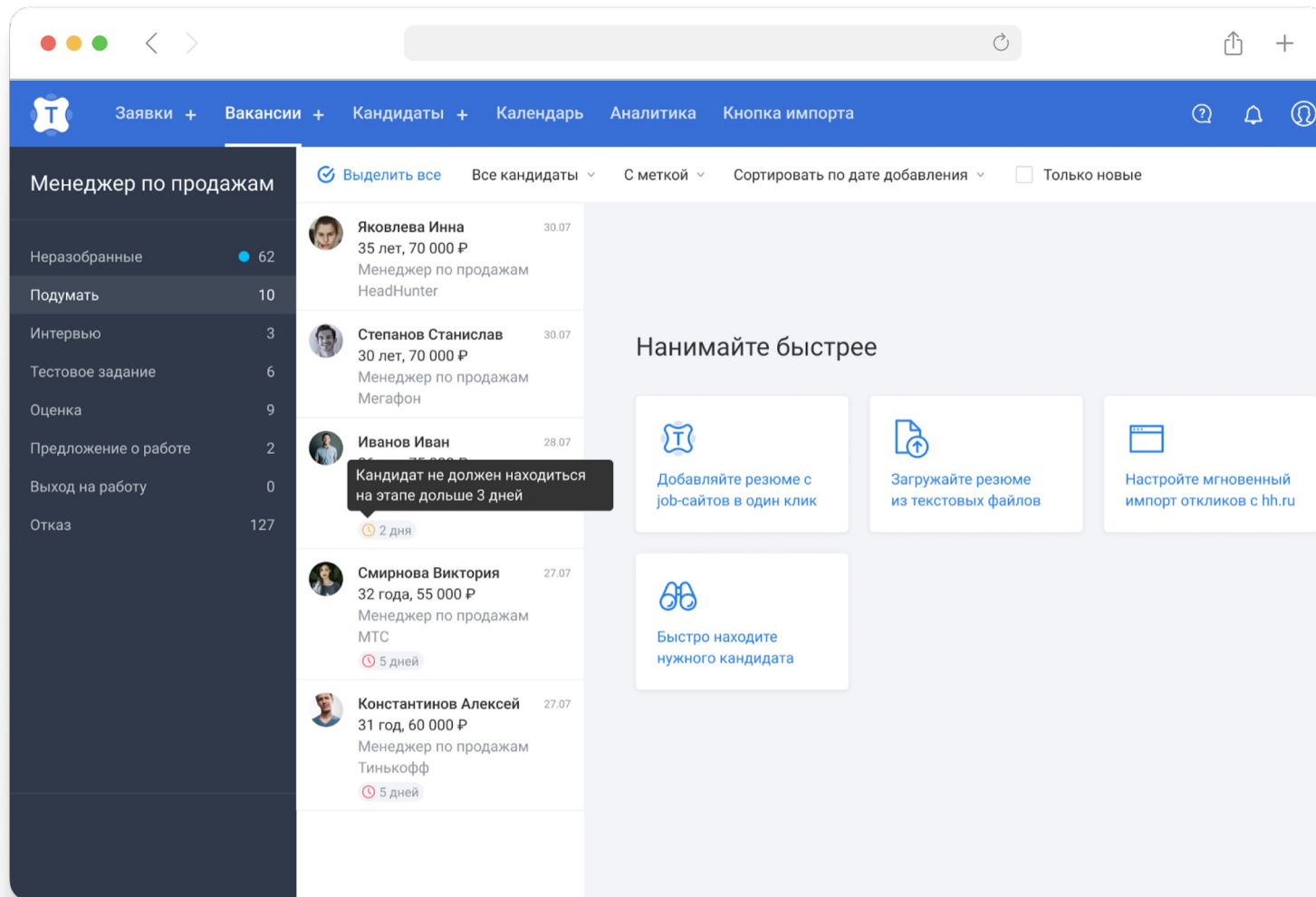
1

АJAX страницы

Одним запросом получаем все данные для страницы

2

Переизбыток данных



Пример трех полей

Запрос на страницу кандидатов ради трех полей

The screenshot displays the HeadHunter interface. On the left is a dark sidebar with filters for search location, vacancies, candidate status, and region. The main area shows a list of candidates, with the first one selected. The right panel shows the detailed profile of 'Масликов 222 Александрович', including his photo, contact information, and work experience.

Список кандидатов:

Имя	Возраст	Профессия	Источник	Дата
Масликов 222	29 лет	Java backend programmer	HeadHunter	02.2022
Масликов Артём	29 лет	Техник 1 категории	НПК СПП	09.2020
Масликов 222	29 лет	Java backend programmer	HeadHunter	02.2022
Масликов Артём	29 лет	Java developer	HeadHunter	02.2020
Масликов 222	29 лет	Java backend programmer	HeadHunter	02.2022
Масликов Артём	29 лет	Java developer	HeadHunter	11.2020
Масликов Артём	29 лет	Java developer	HeadHunter	07.2020

Профиль кандидата: Масликов 222 Александрович
29.07.1994, мужчина, Техник 1 категории, НПК СПП

Источники: Отклик hh x

Метки: [Добавить](#)

Гражданство: Россия
Регион: Москва
Телефон: 8 (905) 123-12-13
WhatsApp Viber Telegram
Домашний телефон: 8 (909) 589-65-95
Email: temaslikov@gmail.com

Резюме hh Резюме из файла Комментарии и история: 4 Письма: 1 Файлы: 2 В вакансиях: 9 Оценки: 0

Java developer
28 сентября 2014, [посмотреть на HeadHunter](#) [Скачать резюме](#)

Опыт работы 2 года 6 месяцев
Июль 2017 — по настоящее время (6 лет 9 месяцев)
HeadHunter

Проблемы текущего подхода

1

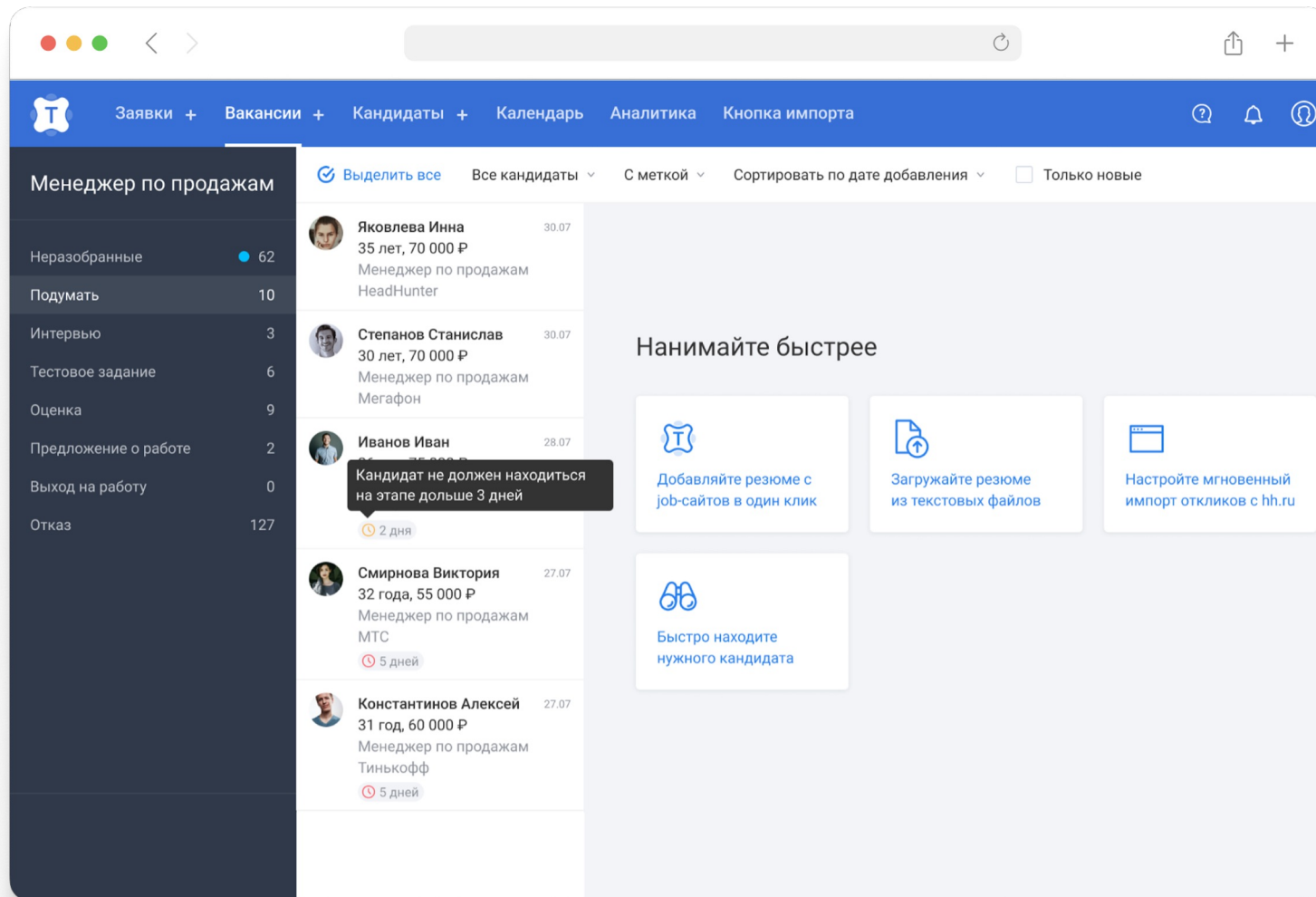
АJAX страницы

Одним запросом получаем все данные для страницы

2

Переизбыток данных

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



Проблемы текущего подхода

1

АJAX страницы

Одним запросом получаем все данные для страницы

2

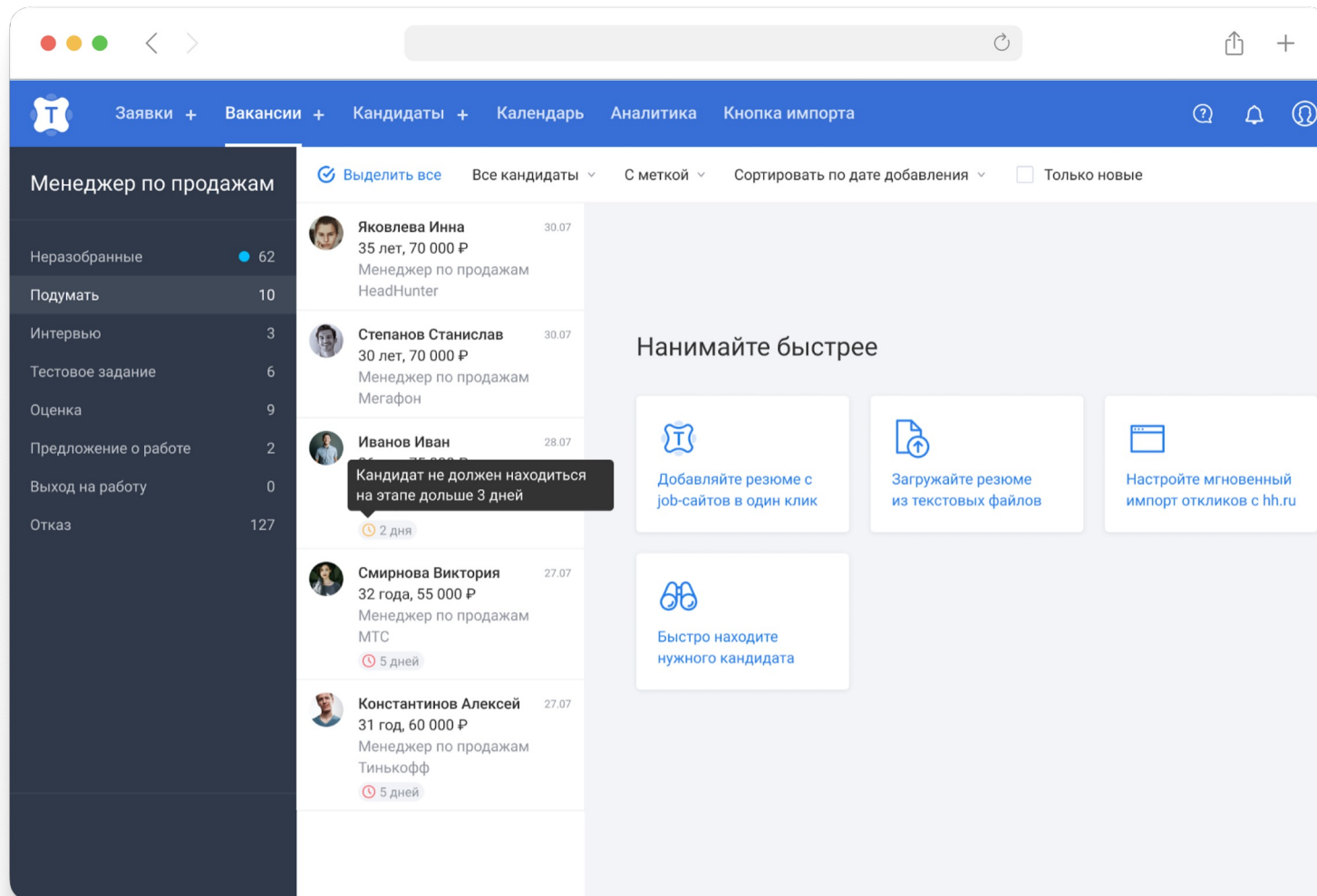
Переизбыток данных

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

3

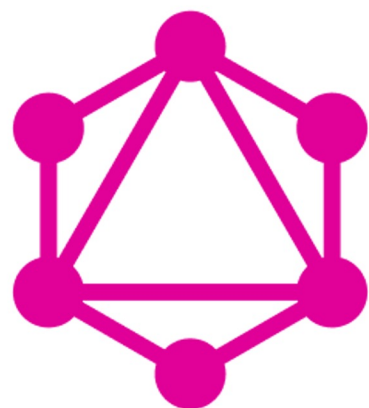
Водопад данных

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



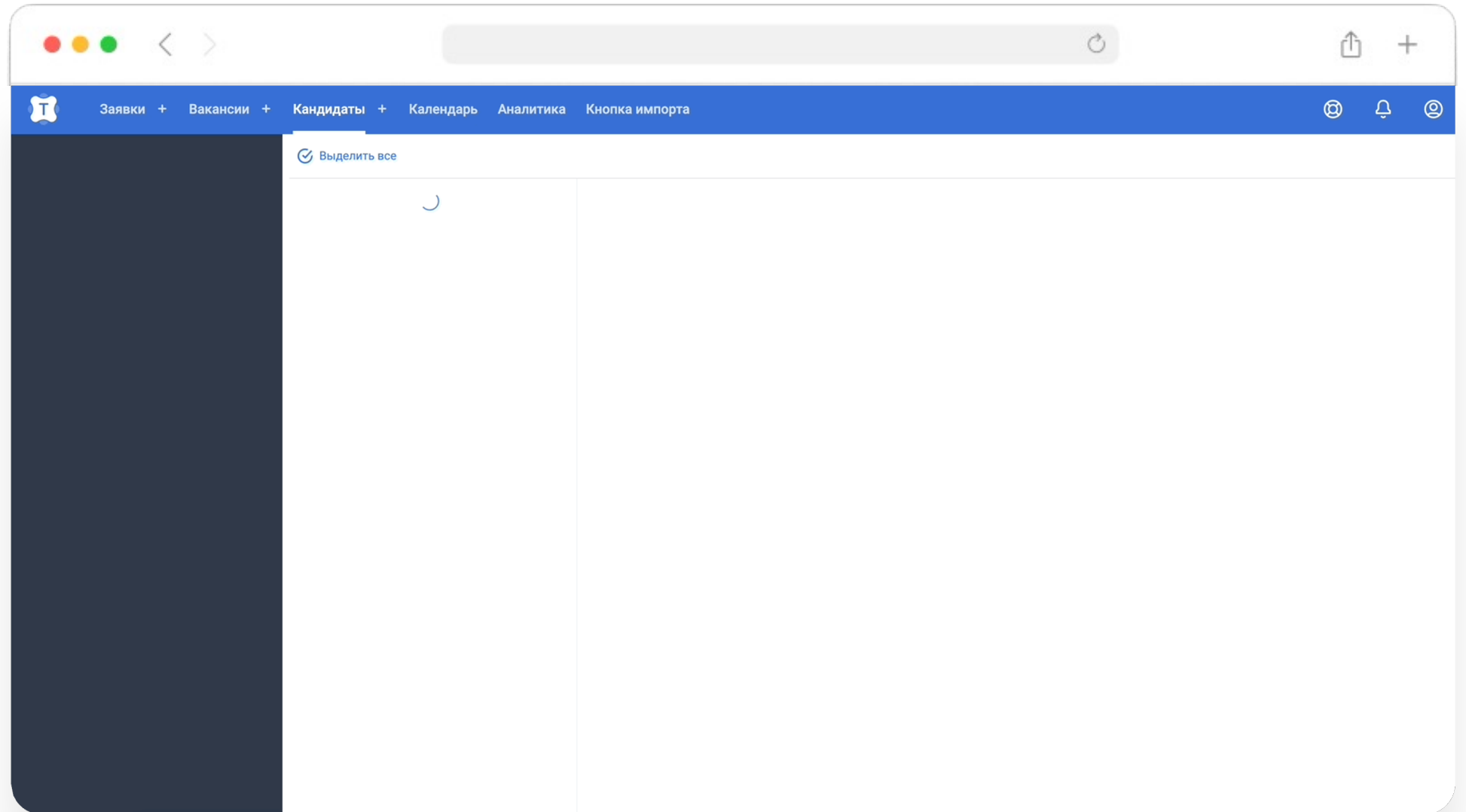
Похоже нам нужен...





GraphQL

Пробуем на странице кандидатов



Ресерч или приключение на 20 минут



С чего начинается ресерч?

С чего начинается ресерч?



С докладов на HolyJS!

graphql holy js

Конференция для JavaScript-разработчиков

15 апреля ONLINE 26-27 апреля МОСКВА + ONLINE

HolyJS

@HolyJS · 27,8 тыс. подписчиков · 586 видео

HolyJS — первая и пока единственная профессиональная конференция по JavaScript... >

cutt.ly/CwklscW и ещё 10 ссылок

Вы подписаны

Главная Видео Трансляции Подкасты Плейлисты Сообщество

Тяжелое утро с HolyJS и Lead IT Project Manager в Samokat.tech

#59

HolyJS · 670 просмотров · Трансляция закончилась 4 недели назад

О конференции — <https://bit.ly/3NQMcqw> Купить билет — <https://bit.ly/3I5Q0Z9> Новый выпуск тяжёлого утра, в котором мы будем...

Для вас

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

- Strong influence on the architecture of the backend and frontend, difficult to start work
- Small community
- Suitable only for React

Фреймворк

Relay



☆ 18.2k



Преимущества

- Мощный инструмент (компилятор, фрагменты, стор)



Недостатки

- Strong influence on the architecture of the backend and frontend, difficult to start work
- Small community
- Suitable only for 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

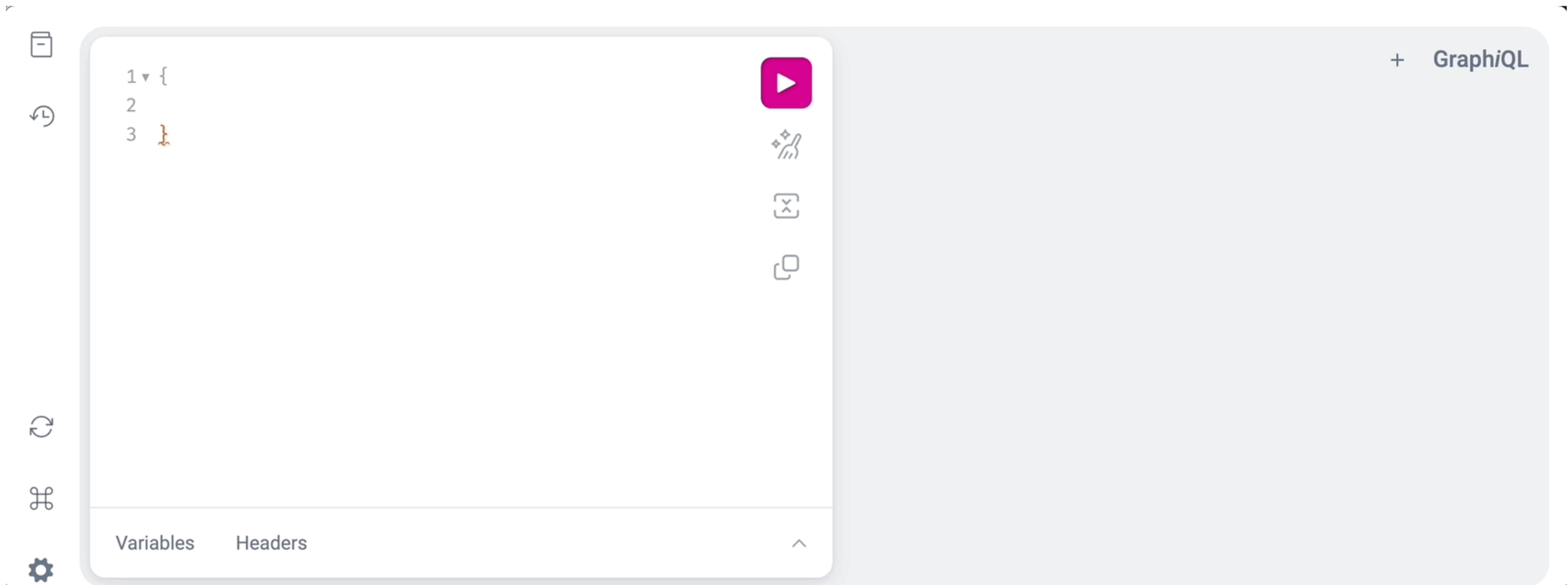


Недостатки

- Маленькое, но дружное комьюнити

Фи́чи

Фича: плейграунд



Фича: типы на схему

```
types.ts M x | schema.graphql M x |
src > __generated__ > types.ts > Country | schema.graphql > ...
20 export type Continent = {
21   __typename?: 'Continent';
22   code: Scalars['ID'];
23   countries: Array<Country>;
24   name: Scalars['String'];
25 };
26
27 export type ContinentFilterInput = {
28   code?: Maybe<StringQueryOperatorInput>;
29 };
30
31 export type Country = {
32   __typename?: 'Country';
33   awsRegion: Scalars['String'];
34   capital?: Maybe<Scalars['String']>;
35   code: Scalars['ID'];
36   continent: Continent;
37   currencies: Array<Scalars['String']>;
38   currency?: Maybe<Scalars['String']>;
39   emoji: Scalars['String'];
40   emojiU: Scalars['String'];
41   languages: Array<Language>;
42   name: Scalars['String'];
43   native: Scalars['String'];
44   phone: Scalars['String'];
45   phones: Array<Scalars['String']>;
46   states: Array<State>;
47   subdivisions: Array<Subdivision>;
48 };
49
50
51 export type CountrynameArgs = {
52   lang?: Maybe<Scalars['String']>;
53 };
1 directive @cacheControl(maxAge: Int, scope: CacheControlScope) on FIELD_DEFINITION
2
3 type Continent {
4   code: ID!
5   countries: [Country!]!
6   name: String!
7 }
8
9 input ContinentFilterInput {
10   code: StringQueryOperatorInput
11 }
12
13 type Country {
14   awsRegion: String!
15   capital: String
16   code: ID!
17   continent: Continent!
18   currencies: [String!]!
19   currency: String
20   emoji: String!
21   emojiU: String!
22   languages: [Language!]!
23   name(lang: String): String!
24   native: String!
25   phone: String!
26   phones: [String!]!
27   states: [State!]!
28   subdivisions: [Subdivision!]!
29 }
30
31 input CountryFilterInput {
32   code: StringQueryOperatorInput
33   continent: StringQueryOperatorInput
34   currency: StringQueryOperatorInput
```

Фича: кодогенерация хуков

```
7 export type CountryListVariables = Types.Exact<{
8   filter?: Types.Maybe<Types.CountryFilterInput>;
9 }>;
10
11
12 export type CountryList = { __typename: 'Query', countries: Array<
13   { __typename: 'Country', code: string, name: string }
14   & CountryCardCountry
15 }> };
16
17
18 export const CountryListDocument: DocumentNode = {"kind": "Document", "definitions": [{"kind": "OperationDefinit
19
20 /**
21  * __useCountryList__
22  *
23  * To run a query within a React component, call `useCountryList` and pass it any options that fit your need
24  * When your component renders, `useCountryList` returns an object from Apollo Client that contains loading,
25  * you can use to render your UI.
26  *
27  * @param baseOptions options that will be passed into the query, supported options are listed on: 
```


Фича: моки для тестов

```
CountriesPage.test.tsx M X
src > pages > CountriesPage > CountriesPage.test.tsx > test('shows error indication if request failed') callback
1 > import '@testing-library/jest-dom/extend-expect';
7
Run | Debug | View story | Profile | Focus
8 test('shows loading indication while loading', async () => { 51ms
9   renderWithProviders(
10     <ApolloMockProvider>
11     | <CountriesPage />
12     </ApolloMockProvider>,
13   );
14
15   const loadingIndicator = await screen.findByText('Loading...');
16   expect(loadingIndicator).toBeDefined();
17 });
18
Run | Debug | View story | Profile | Focus
19 test('shows error indication if request failed', async () => { 41ms
20   renderWithProviders(
21     <ApolloMockProvider errors={{ message: 'Fail!' }}>
22     | <CountriesPage />
23     </ApolloMockProvider>,
24   );
25
26   const errorMessage = await screen.findByText('Fail!');
27   expect(errorMessage).toBeDefined();
28 });
29
Run | Debug | View story | Profile | Focus
30 test('shows loaded countries', async () => { 41ms
31   const queryMocks = {
32     countries: () => [{ name: 'Kazakhstan' }, { name: 'Russia' } ],
33   };
34
35   renderWithProviders(
36     <ApolloMockProvider queryMocks={queryMocks}>
```

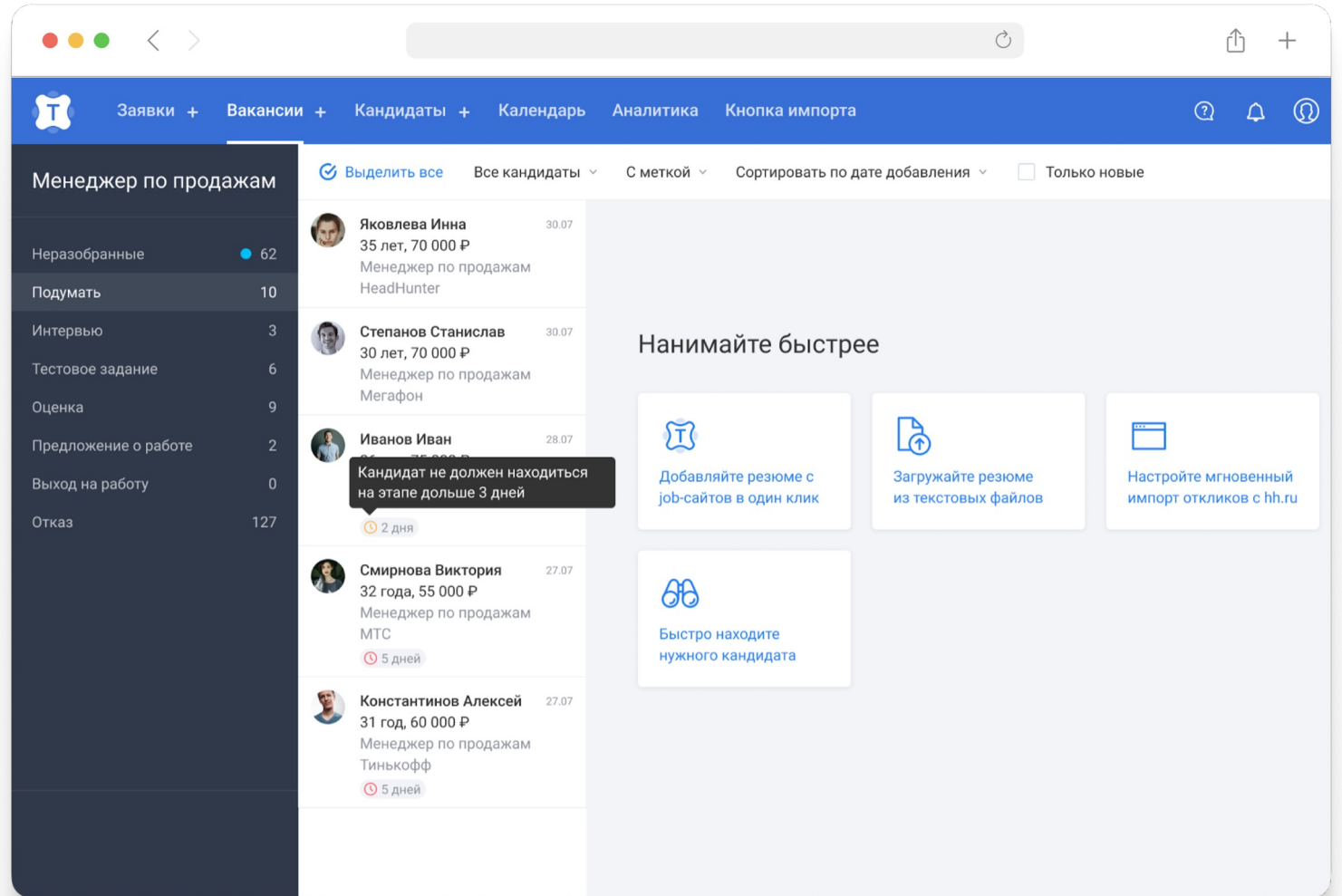

Одним словом

Одним словом



Итоги ресерча

Обсуждаем

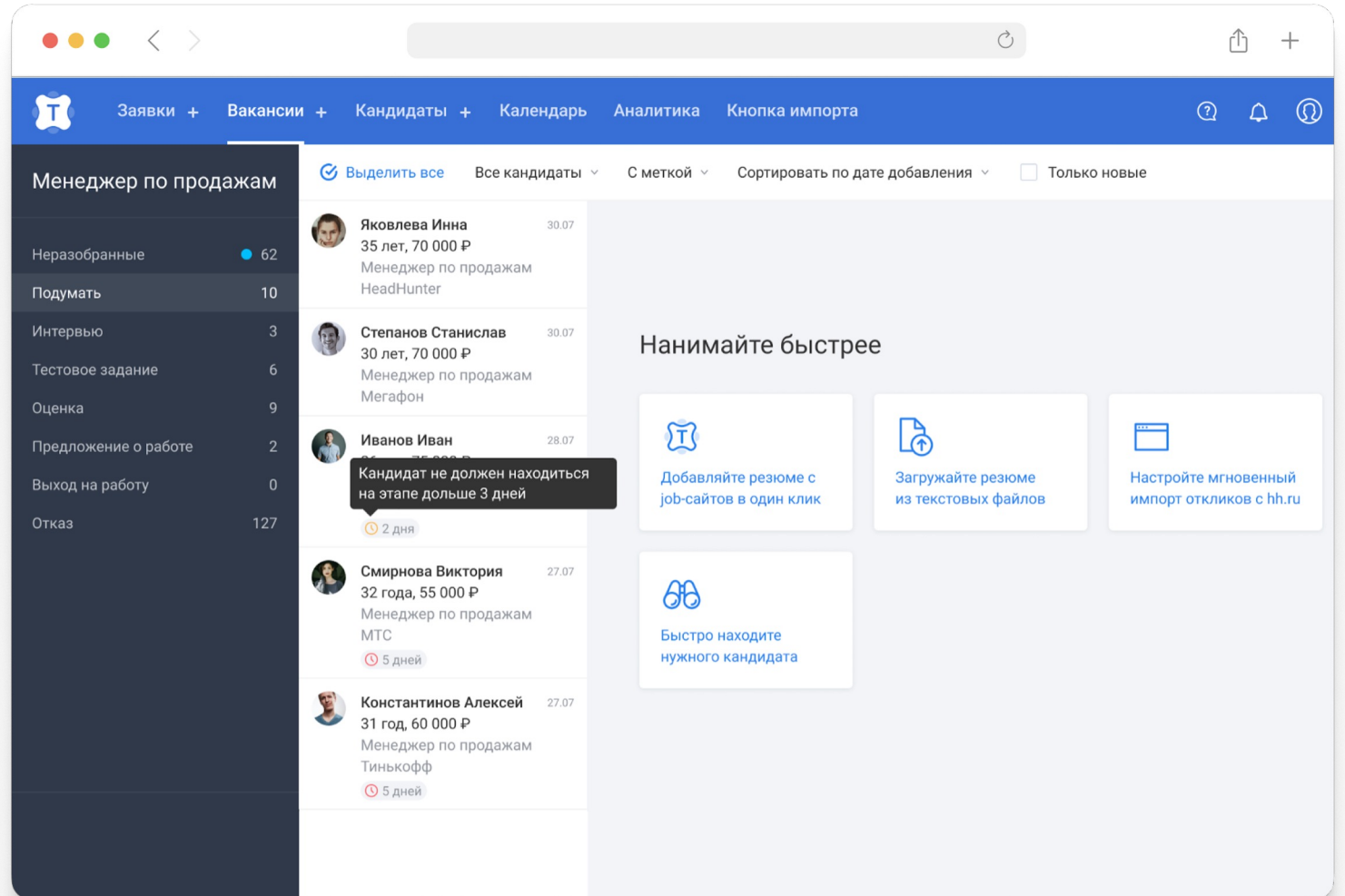


Обсуждаем

1

Бэк

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



Обсуждаем

1

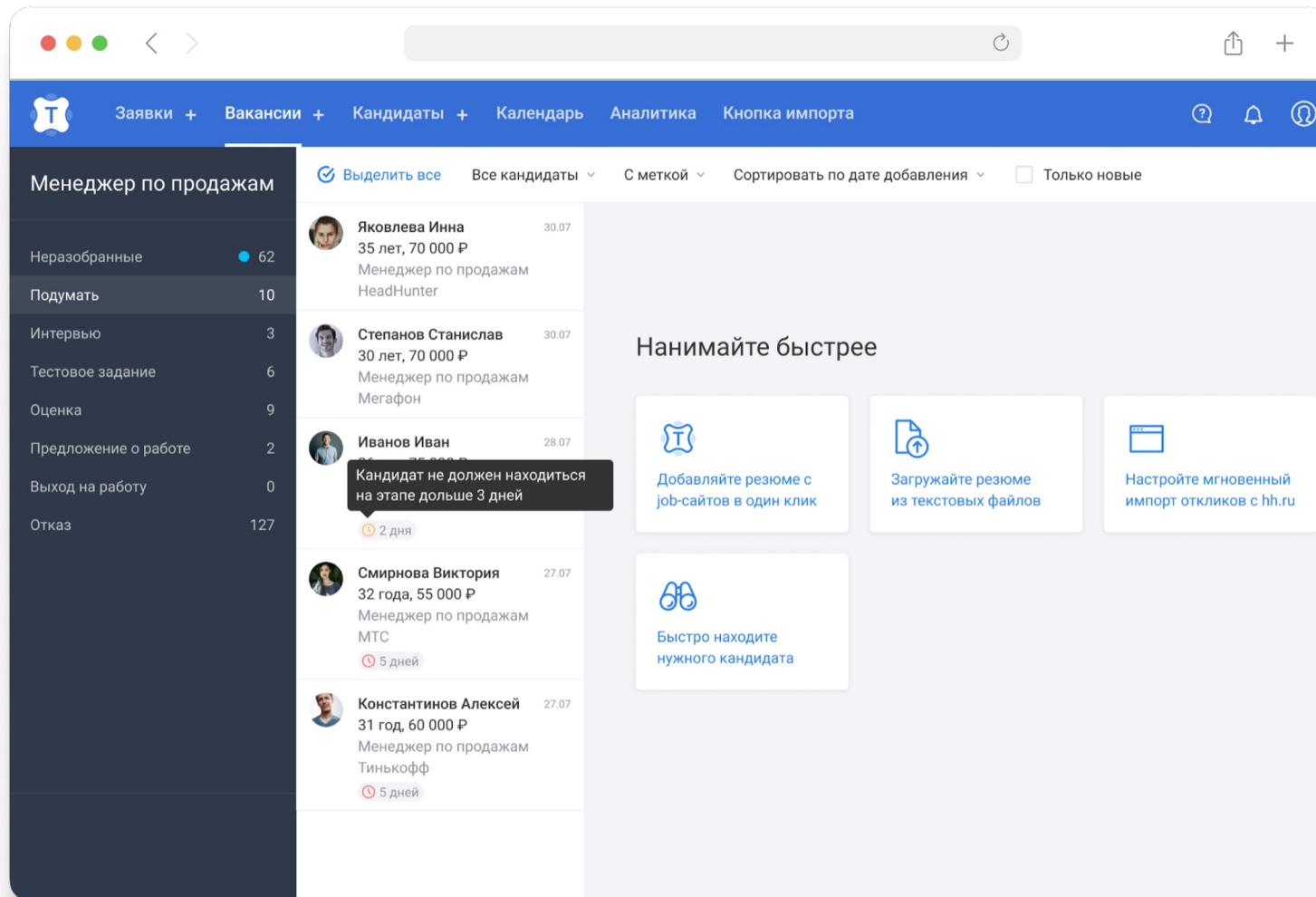
Бэк

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

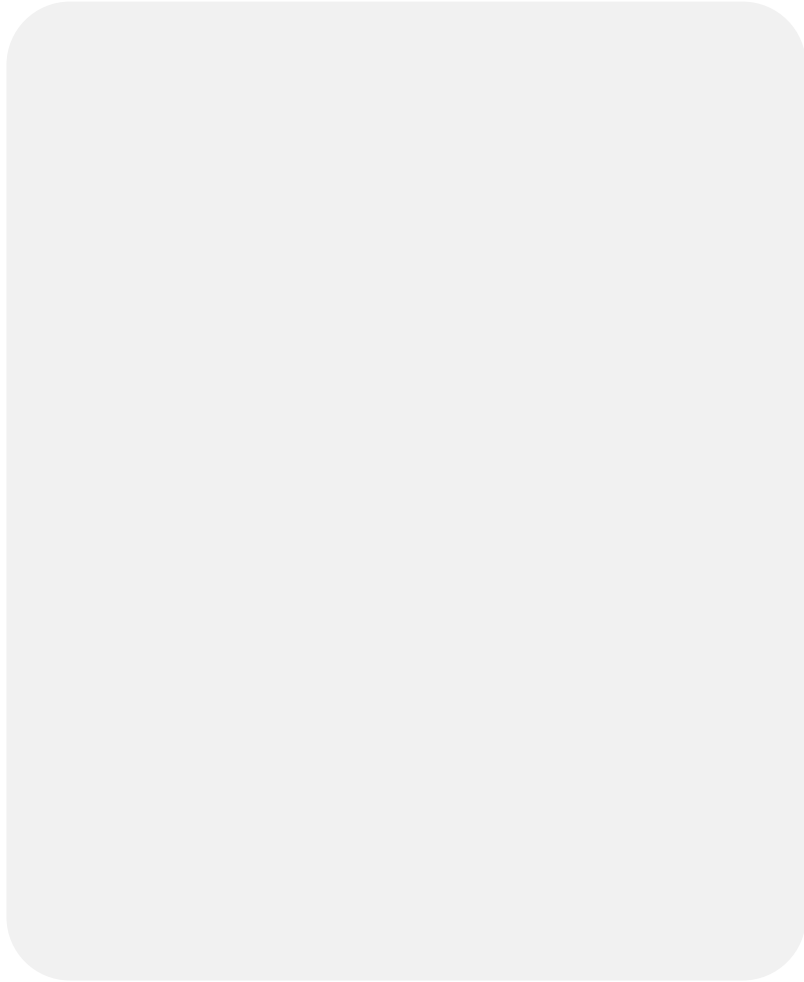
2

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 вошло в frontend оценку?

- Думать о синхронизации кэша фреймворка с **Redux**
- Правильно в **View** слой КОМПОНЕНТОВ

```
vacanciesPage.js x
app > containers > ats > vacanciesPage.js > AtsVacanciesPage
74 class AtsVacanciesPage extends Component {
101   tabnine: test | explain | document | ask
102   componentDidMount() {
103     this.loadVacancies();
104   }
105   Nikita Mostovoy, 8 years ago • HH-611488 add my vacancies page
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
```

Что вошло в frontend оценку?

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

```
vacanciesPage.js x
app > containers > ats > vacanciesPage.js > AtsVacanciesPage
74 class AtsVacanciesPage extends Component {
101   tabnine: test | explain | document | ask
102   componentDidMount() {
103     this.loadVacancies();
104   }
105   Nikita Mostovoy, 8 years ago • HH-611488 add my vacancies page
106   tabnine: test | explain | document | ask
107   componentDidUpdate(prevProps) {
108     if (
109       this.props[HH_BACKURL_PUBLISHED_PARAMETER] !== prevProps[HH_BACKURL_PUBLISHED_PARAMETER] ||
110       this.props.accountHhVacancyImportSettings !== prevProps.accountHhVacancyImportSettings
111     ) {
112       this.handlePublishedHHId();
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
```

Что вошло в frontend оценку?

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

```
vacanciesPage.js x
app > containers > ats > vacanciesPage.js > AtsVacanciesPage
74 class AtsVacanciesPage extends Component {
101   tabnine: test | explain | document | ask
102   componentDidMount() {
103     this.loadVacancies();
104   }
105   Nikita Mostovoy, 8 years ago • HH-611488 add my vacancies page
106   tabnine: test | explain | document | ask
107   componentDidUpdate(prevProps) {
108     if (
109       this.props[HH_BACKURL_PUBLISHED_PARAMETER] !== prevProps[HH_BACKURL_PUBLISHED_PARAMETER] ||
110       this.props.accountHhVacancyImportSettings !== prevProps.accountHhVacancyImportSettings
111     ) {
112       this.handlePublishedHHId();
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
```

Что вошло 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**

```
vacanciesPage.js x
app > containers > ats > vacanciesPage.js > AtsVacanciesPage
74 class AtsVacanciesPage extends Component {
101   tabnine: test | explain | document | ask
102   componentDidMount() {
103     this.loadVacancies();
104   }
105   Nikita Mostovoy, 8 years ago • HH-611488 add my vacancies page
106   tabnine: test | explain | document | ask
107   componentDidUpdate(prevProps) {
108     if (
109       this.props[HH_BACKURL_PUBLISHED_PARAMETER] !== prevProps[HH_BACKURL_PUBLISHED_PARAMETER] ||
110       this.props.accountHhVacancyImportSettings !== prevProps.accountHhVacancyImportSettings
111     ) {
112       this.handlePublishedHHId();
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
```

Оценка: больше одного месяца с обеих сторон



Все, что могли — отрезали



Все, что могли — отрезали



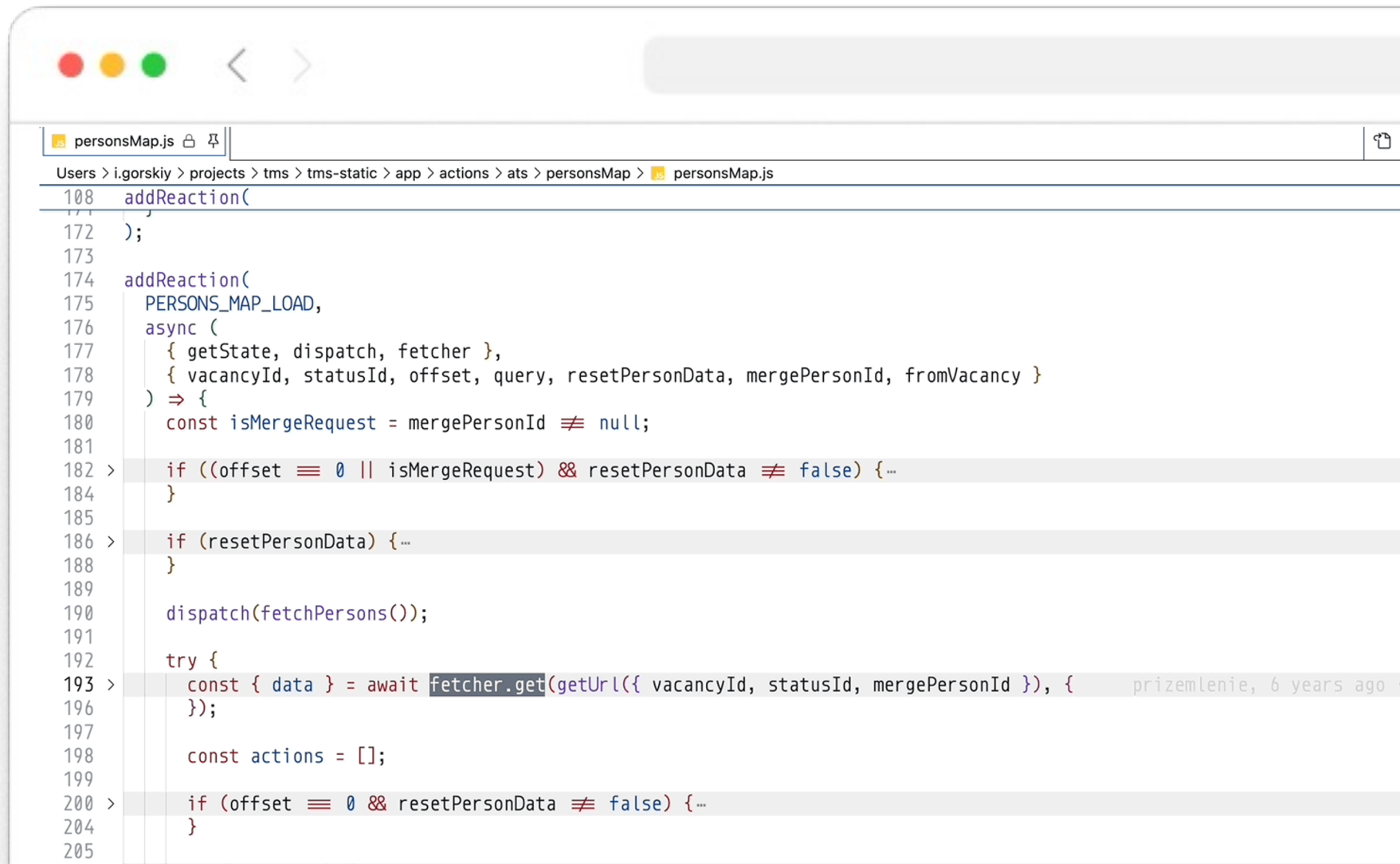
Челлендж

Челлендж



Помните асинхронные экшены?

Помните асинхронные экшены?



```
personsMap.js
Users > i.gorskiy > projects > tms > tms-static > app > actions > ats > personsMap > personsMap.js
108  addReaction(
171  );
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
```

Напишем свой fetcher и подменим его!



Важно!

Работаем только на уровне сети

Важно!

Не трогаем слой `view`

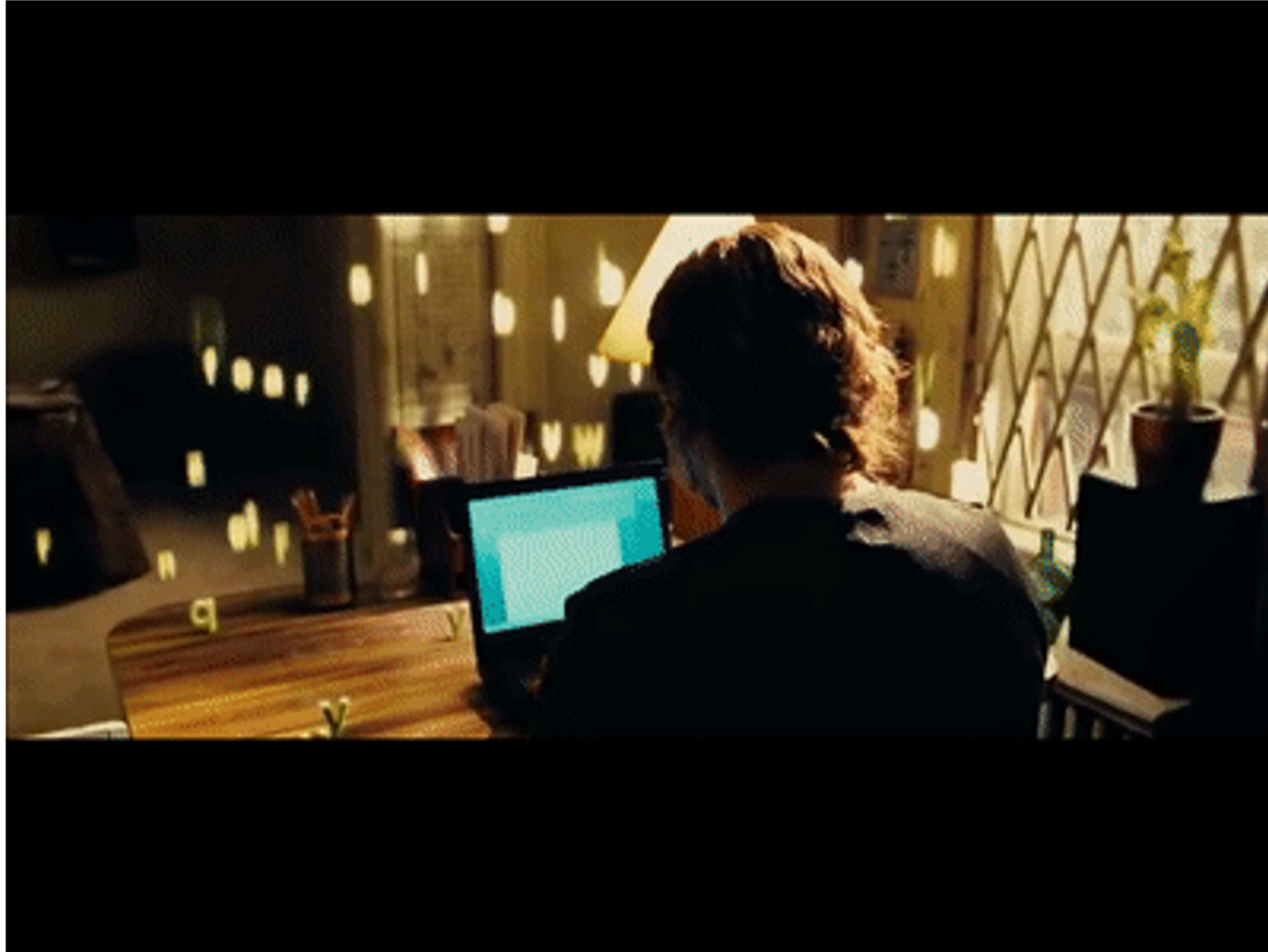


**Получается
сильно дешевле
внедрения
фреймворка**

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

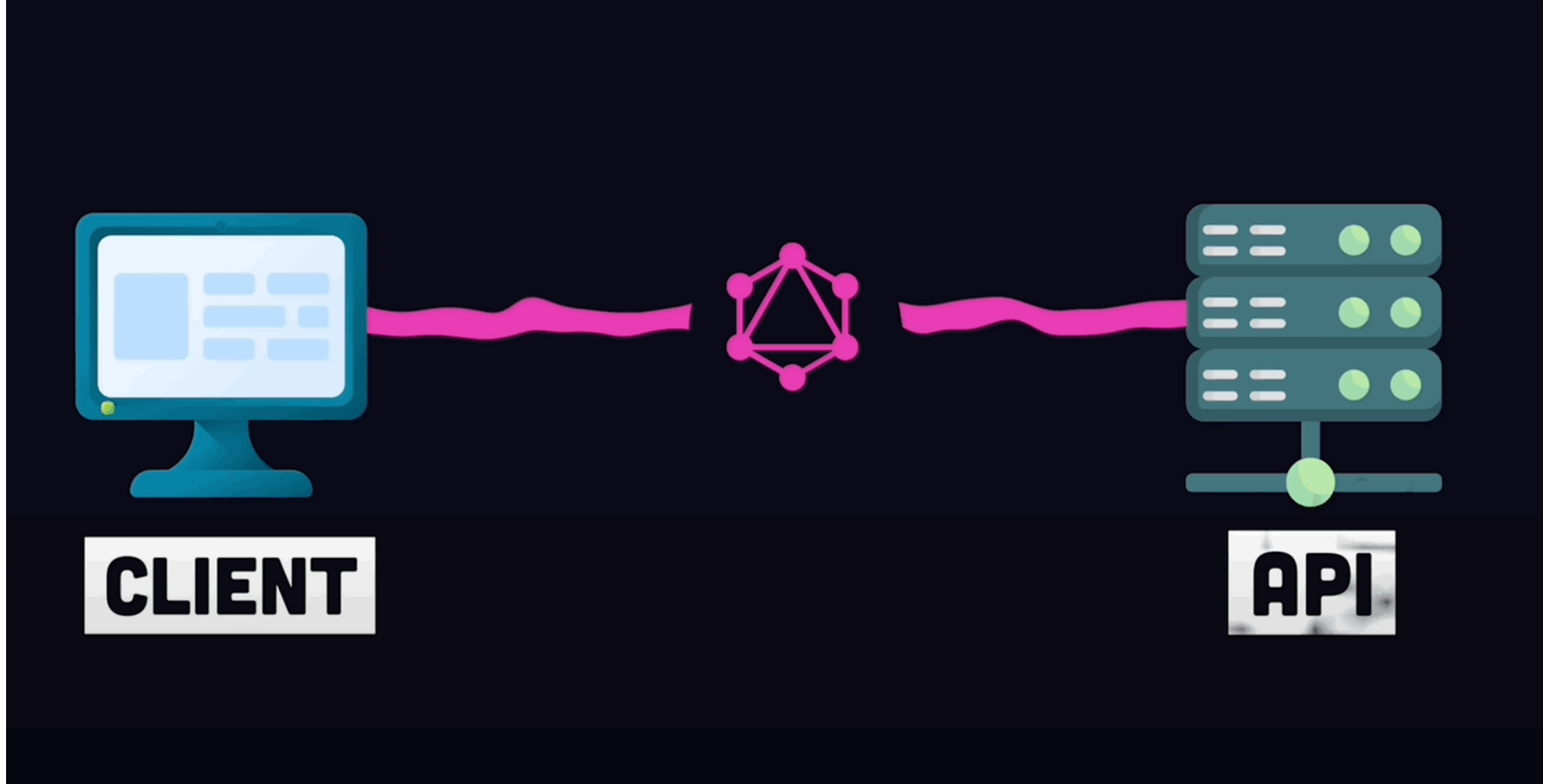
- Получение схемы
- Генерация типов по схеме
- Генерация fetcher функций под каждый запрос/мутацию
- Генерация фрагментов



04.

Тесты

- Мок операций
- Генерация стабов для тестов



Клиент: контракт GraphQL response

```
1 /**
2  * @see https://spec.graphql.org/October2021/#sec-Errors.Error-result-format
3  */
4 interface Error {
5     message: string
6     column: { line: number; column: number }[]
7     path: string[]
8     [x: string]: unknown
9 }
10 /**
11  * @see https://spec.graphql.org/October2021/#sec-Errors
12  */
13 interface FailResponse<T> {
14     data: T | null
15     errors: Error[]
16 }
17 /**
18  * @see https://spec.graphql.org/October2021/#sec-Response
19  */
20 interface SuccessResponse<T> {
21     data: T
22     errors: []
23 }
24
25 export type Response<T extends object> = SuccessResponse<T> | FailResponse<T>
```

Клиент: контракт GraphQL response

```
1 /**
2  * @see https://spec.graphql.org/October2021/#sec-Errors.Error-result-format
3  */
4  interface Error {
5      message: string
6      column: { line: number; column: number }[]
7      path: string[]
8      [x: string]: unknown
9  }
10 /**
11  * @see https://spec.graphql.org/October2021/#sec-Errors
12  */
13 interface FailResponse<T> {
14     data: T | null
15     errors: Error[]
16 }
17 /**
18  * @see https://spec.graphql.org/October2021/#sec-Response
19  */
20 interface SuccessResponse<T> {
21     data: T
22     errors: []
23 }
24
25 export type Response<T extends object> = SuccessResponse<T> | FailResponse<T>
```

Клиент: request

```
1 import { type DocumentNode } from 'graphql'
2
3 export type Query = DocumentNode | string
4
5 export type Variables<T extends object> = T
6
7 export interface RequestFn {
8   <D extends object, V extends object>(query: Query variables?: Variables<V>): Promise<Response<D>>
9 }
```

Клиент: request

```
1 import { type ASTNode, type OperationDefinitionNode } from 'graphql'
2 import type { RequestFn } from './types'
3
4 const isOperation = (node: ASTNode): node is OperationDefinitionNode => node.kind === 'OperationDefinition'
5 const getOperationName = (node: ASTNode): string => isOperation(node) ? node.name!.value : 'unknown'
6
7 export const request: <U extends string>(url: U) => RequestFn = (url) => async (query, variables) => {
8   const isDocumentNode = typeof query !== 'string';
9   const operationName = isDocumentNode ? getOperationName(query.definitions[0]) : 'unknown';
10  const api = new URL(url)
11  const params = {
12    method: 'POST',
13    headers: {
14      'Content-Type': 'application/json',
15    },
16    body: JSON.stringify({
17      query: isDocumentNode ? query.loc?.source.body : query,
18      variables,
19      operationName,
20    }),
21  }
22  api.searchParams.append('operationName', operationName)
23
24  const response = await fetch(`${api}`, params)
25  const data = await response.json()
26
27  return data
28 }
```

Клиент: request

```
1 import { type ASTNode, type OperationDefinitionNode } from 'graphql'
2 import type { RequestFn } from './types'
3
4 const isOperation = (node: ASTNode): node is OperationDefinitionNode => node.kind === 'OperationDefinition'
5 const getOperationName = (node: ASTNode): string => isOperation(node) ? node.name!.value : 'unknown'
6
7 export const request: <U extends string>(url: U) => RequestFn = (url) => async (query, variables) => {
8   const isDocumentNode = typeof query !== 'string';
9   const operationName = isDocumentNode ? getOperationName(query.definitions[0]) : 'unknown';
10  const api = new URL(url)
11  const params = {
12    method: 'POST',
13    headers: {
14      'Content-Type': 'application/json',
15    },
16    body: JSON.stringify({
17      query: isDocumentNode ? query.loc?.source.body : query,
18      variables,
19      operationName,
20    }),
21  }
22  api.searchParams.append('operationName', operationName)
23
24  const response = await fetch(`${api}`, params)
25  const data = await response.json()
26
27  return data
28 }
```


Клиент: AST DocumentNode

AST Explorer Snippet GraphQL graphql-js Transform default Parser: [graphql-js-15.0.0](#)

```
1 query GetUsers {
2   users {
3     id,
4     name,
5   }
6 }
7
```

Tree JSON

Autofocus Hide methods Hide empty keys Hide location data
 Hide type keys

```
  directives: [ ]
  + selectionSet: SelectionSet {kind, selections, loc}
  + loc: {start, end, startToken, endToken, source, ... +2}
}
- loc: {
  start: 0
  end: 51
  + startToken: <SOF> {kind, start, end, line, column, ... +5}
  + endToken: <EOF> {kind, start, end, line, column, ... +5}
  - source: {
    body: "query GetUsers {\n  users {\n    id,\n    name,\n  }\n}\n"
    name: "GraphQL request"
    + locationOffset: {line, column}
  }
  toJSON: (...)
  inspect: (...)
}
```

Клиент: `operationName` ключ к успеху

Клиент: добавляем operationName

```
1 import { type ASTNode, type OperationDefinitionNode } from 'graphql'
2 import type { RequestFn } from './types'
3
4 const isOperation = (node: ASTNode): node is OperationDefinitionNode => node.kind === 'OperationDefinition'
5 const getOperationName = (node: ASTNode): string => isOperation(node) ? node.name!.value : 'unknown'
6
7 export const request: <U extends string>(url: U) => RequestFn = (url) => async (query, variables) => {
8   const isDocumentNode = typeof query !== 'string';
9   const operationName = isDocumentNode ? getOperationName(query.definitions[0]) : 'unknown';
10  const api = new URL(url)
11  const params = {
12    method: 'POST',
13    headers: {
14      'Content-Type': 'application/json',
15    },
16    body: JSON.stringify({
17      query: isDocumentNode ? query.loc?.source.body : query,
18      variables,
19      operationName,
20    }),
21  }
22  api.searchParams.append('operationName', operationName)
23
24  const response = await fetch(`${api}`, params)
25  const data = await response.json()
26
27  return data
28 }
```

Клиент: AST operationName

AST Explorer Snippet GraphQL </> graphql-js Transform default ? Parser: [graphql-js-15.0.0](#)

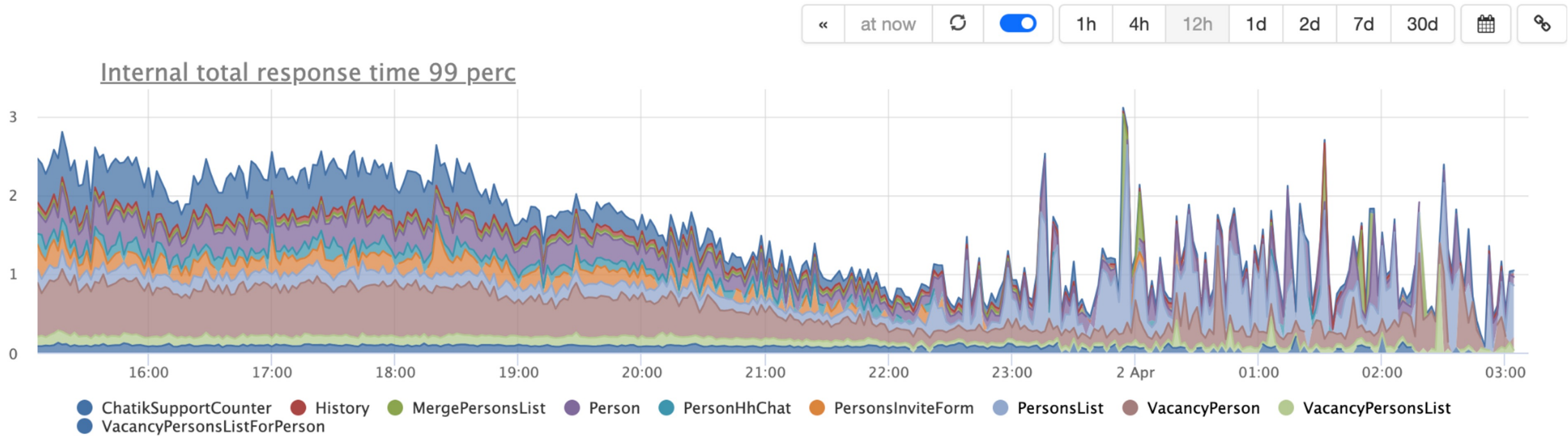
```
1 query GetUsers {
2   users {
3     id,
4     name,
5   }
6 }
7
```

Tree JSON

Autofocus Hide methods Hide empty keys Hide location data
 Hide type keys

```
- Document {
  kind: "Document"
  - definitions: [
    - OperationDefinition {
      kind: "OperationDefinition"
      operation: "query"
      - name: Name {
        kind: "Name"
        value: "GetUsers"
        + loc: {start, end, startToken, endToken, source, ... +2}
      }
      variableDefinitions: [ ]
      directives: [ ]
      + selectionSet: SelectionSet {kind, selections, loc}
      + loc: {start, end, startToken, endToken, source, ... +2}
    }
  ]
  + loc: {start, end, startToken, endToken, source, ... +2}
```

Клиент: operationName для мониторинга



Клиент: operationName в URL для отладки

The screenshot displays a web application interface for managing candidates. The top navigation bar includes links for 'Заявки', 'Вакансии', 'Кандидаты', 'Календарь', 'Аналитика', and 'Кнопка импорта'. The main content area shows a list of candidates with details such as age, salary, and job title. A search bar is visible on the left side. The browser's developer tools are open, showing the network tab with a request to the GraphQL endpoint. The request URL includes 'operationName=' followed by a query name, demonstrating how to debug GraphQL queries by identifying the operation name in the URL.

Название	Статус	Тип	Инициатор	Размер	Время	Каскад загрузки

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 X
client > ts 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
```

```
simple-graphql-bootstrap on main [x!+?] via v16.15.0
● → nvm use
Found '/Users/i.gorskiy/projects/simple-graphql-bootstrap/.nvmrc' with version <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

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

```
1 import path from 'node:path'
2 import fs from 'node:fs'
3 import {
4     getIntrospectionQuery,
5     buildClientSchema,
6     IntrospectionQuery,
7 } from 'graphql'
8 import { Fetcher } from '../client/fetcher'
9
10 async function fetchSchema(url: string) {
11     const fetcher = new Fetcher(url)
12     const { data } = await fetcher.query<IntrospectionQuery>(
13         getIntrospectionQuery()
14     )
15
16     if (data) {
17         console.log('✅', 'download schema introspection')
18         return buildClientSchema(data)
19     }
20 }
```


Кодогенерация: получаем schema.graphql

```
1 import {
2     getIntropectionQuery,
3     buildClientSchema,
4     IntrospectionQuery,
5 } from 'graphql'
6 import { Fetcher } from '../../client/fetcher'
7
8 async function fetchSchema(url: string) {
9     const fetcher = new Fetcher(url)
10    const { data } = await fetcher.query<IntrospectionQuery>(
11        getIntrospectionQuery()
12    )
13
14    if (data) {
15        console.log('✅', 'download schema introspection')
16        return buildClientSchema(data)
17    }
18 }
```

Кодогенерация: получаем schema.GraphQL

```
1 function printClientSchema(data: GraphQLSchema): string {
2     const schema = printSchema(data)
3     console.log('✅', 'print client schema')
4     return schema
5 }
6
7 function writeSchema(dist: string, schema: string) {
8     const output = path.resolve(process.cwd(), dist)
9     console.log('✅', 'write schema file', dist)
10    fs.writeFileSync(output, schema)
11 }
12
13 function printFile(path: string) {
14     return (schema?: GraphQLSchema) => {
15         !fs.existsSync(path) && fs.mkdirSync(path)
16         schema && writeSchema(`${path}/schema.graphql`, printClientSchema(schema))
17     }
18 }
19
20 fetchSchema('https://rickandmortyapi.com/graphql')
21     .then(printFile('./codegen'))
22     .then(() => console.log('✅', 'success create schema.graphql'))
23     .catch((e) => console.error('❌', 'failed create schema.graphql', 'error:', e))
```

Кодогенерация: получаем schema.graphql

```
1 function printClientSchema(data: GraphQLSchema): string {
2   const schema = printSchema(data)
3   console.log('✅', 'print client schema')
4   return schema
5 }
6
7 function writeSchema(dist: string, schema: string) {
8   const output = path.resolve(process.cwd(), dist)
9   console.log('✅', 'write schema file', dist)
10  fs.writeFileSync(output, schema)
11 }
12
13 function printFile(path: string) {
14   return (schema?: GraphQLSchema) => {
15     !fs.existsSync(path) && fs.mkdirSync(path)
16     schema && writeSchema(`${path}/schema.graphql`, printClientSchema(schema))
17   }
18 }
19
20 fetchSchema('https://rickandmortyapi.com/graphql')
21   .then(printFile('./codegen'))
22   .then(() => console.log('✅', 'success create schema.graphql'))
23   .catch((e) => console.error('❌', 'failed create schema.graphql', 'error:', e))
```

Кодогенерация: получаем `schema.graphql`



Кодогенерация: получаем schema.graphql

```
schema.ts M x | schema.graphql M | node + v | ... x
scripts > schema > schema.ts > fetchSchema
1 > import path from 'node:path'...
12
13 const fetcher = new Fetcher('https://rickandmortyapi.com/graphql')
14
15 async function fetchSchema() {
16   const { data } = await fetcher.query<
17     IntrospectionQuery,
18     { operationName: 'IntrospectionQuery' }
19   >(getIntrospectionQuery(), { operationName: 'IntrospectionQuery' })
20
21   if (data) {
22     console.log('✅', 'download schema introspection')
23     return buildClientSchema(data)
24   }
25 }
26
27 function printClientSchema(data: GraphQLSchema): string {
28   const schema = printSchema(data)
29   console.log('✅', 'print client schema')
30   return schema
31 }
32
33 function writeSchema(dist: string, schema: string) {
34   const output = path.resolve(process.cwd(), dist)
35   fs.writeFileSync(output, schema)
36   console.log('✅', 'write schema file', dist)
37 }
38
39 function printFile(schema?: GraphQLSchema) {
40   if (!fs.existsSync('./.codegen')) {
41     fs.mkdirSync('./.codegen')
42   }
43   if (schema) {
44     writeSchema('./.codegen/schema.graphql', printClientSchema(schema))
45   }
46 }
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
simple-graphql-bootstrap on main [x!+?] via v18.18.2
├─ yarn schema
├─ yarn run v1.22.21
└─ npx ts-node ./scripts/schema/schema.ts
```

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** - plain string; for an array of strings, use an array if you're using a `gql` tag or a string instead of an array [here](#)
- **generates** (optional) - represents a list of files to generate. Each item can be specified:
 - **generate: { plugin: 'plugin-name', filename: 'path/to/output/file' }** - plugins, considered as plugins, also point to the output file [here](#). You can manually create the list of output files
 - **generate: { plugin: 'plugin-name', filename: 'path/to/output/file' }** - plugin file
 - **generate: { plugin: 'plugin-name', filename: 'path/to/output/file' }** - plugin file
 - **generate: { plugin: 'plugin-name', filename: 'path/to/output/file' }** - plugin file
- **require** - A list of plugins to require. This option is essential for some plugins that require unsupported format [\(more information\)](#). Note that values specified in your `.yaml` file get loaded after loading the `.yml` file
- **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

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

Кодогенерация: добавляем типизированную схему

```
1  schema: ".codegen/schema.graphql"
2  documents:
3    - "./src/**/*.graphql"
4  ignoreNoDocuments: true
5  extensions:
6    codegen:
7      errorsOnly: true
8      generates:
9        ./types/schema.ts:
10         config:
11           defaultScalarType: unknown
12           scalars:
13             ID: "number"
14         plugins:
15           - typescript
16           - add:
17             content: "/* eslint-disable */\n\n /* Auto generated file */ \n "
```

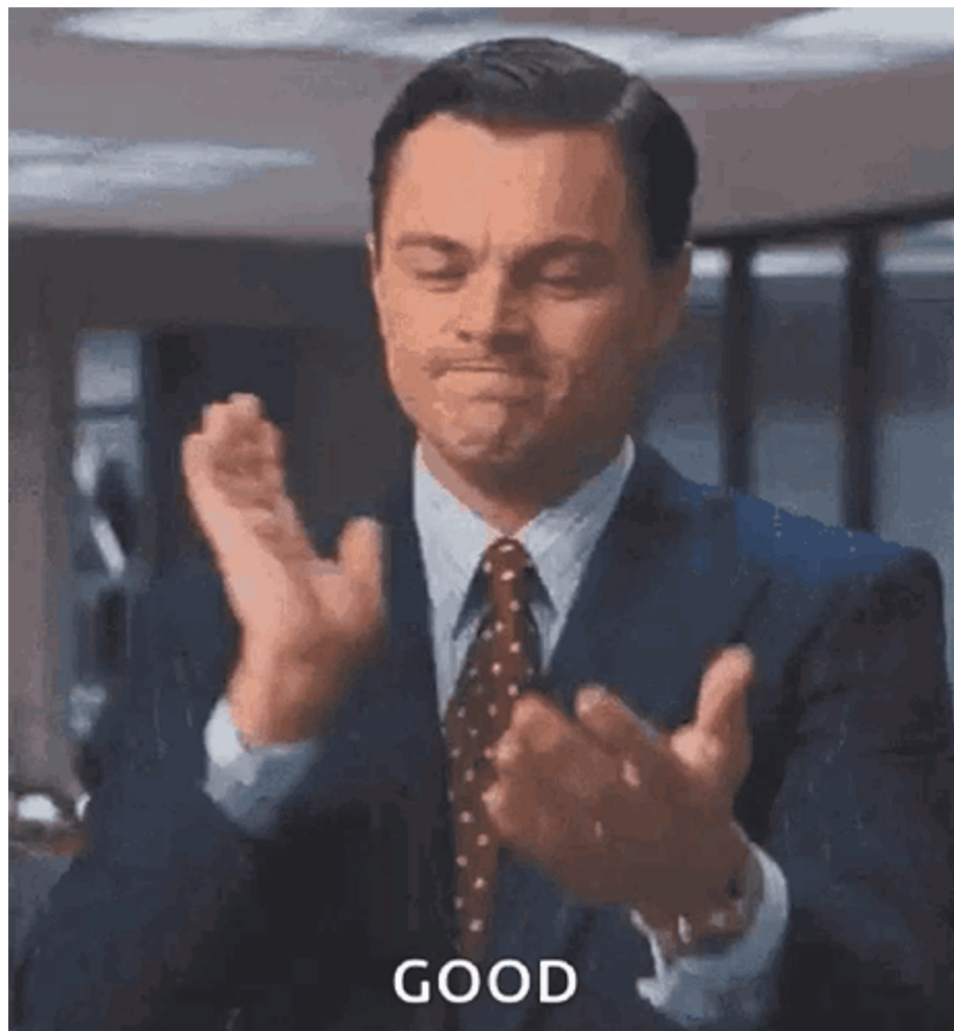
Кодогенерация: добавляем типизированную схему

```
1  schema: ".codegen/schema.graphql"
2  documents:
3    - "./src/**/*.graphql"
4  ignoreNoDocuments: true
5  extensions:
6    codegen:
7      errorsOnly: true
8      generates:
9        ./types/schema.ts:
10         config:
11           defaultScalarType: unknown
12           scalars:
13             ID: "number"
14         plugins:
15           - typescript
16           - add:
17             content: "/* eslint-disable */\n\n /* Auto generated file */ \n "
```

Кодогенерация: смотрим типизированную схему

```
package.json M x schema.ts M
package.json > {} scripts > graphql:generate
1 {
2   "name": "simple-graphql-bootstrap",
3   "version": "1.0.0",
4   "license": "MIT",
5   "type": "module",
6   "engines": {
7     "node": "≥ v18.18.2"
8   },
9   > Debug
10  "scripts": {
11    "schema": "npx ts-node ./scripts/schema/schema.ts",
12    "build-plugin": "tsc -p ./scripts/bootstrap-generate/tsconfig.json",
13    "build": "tsc -p ./tsconfig.json",
14    "test": "jest",
15    "lint": "eslint",
16    "ts-check": "tsc -p ./tsconfig.json --noEmit",
17    "graphql:generate": "graphql-codegen-esm ./graphqlrc.yml",
18    "graphql:generate-watch": "graphql-codegen-esm --watch ./graphqlrc.yml"
19  },
20  "dependencies": {
21    "@babel/preset-env": "^7.24.3",
22    "@babel/preset-typescript": "^7.24.1",
23    "graphql": "^16.8.1",
24    "graphql-tag": "^2.12.6",
25    "prettier": "^3.2.5",
26    "typescript": "^5.4.3",
27    "undici": "^6.10.1"
28  },
29  "devDependencies": {
30    "@babel/core": "^7.24.3",
31    "@graphql-codegen/cli": "^5.0.2",
32    "@graphql-codegen/near-operation-file-preset": "^3.0.0",
33    "@graphql-codegen/typescript": "^4.0.6",
34    "@graphql-eslint/eslint-plugin": "^3.20.1",
```

```
simple-graphql-bootstrap on main [x!+?] via v18.18.2
```



Кодогенерация шаблонного кода

Фича: кодогенерация хуков

```
7 export type CountryListVariables = Types.Exact<{
8   filter?: Types.Maybe<Types.CountryFilterInput>;
9 }>;
10
11
12 export type CountryList = { __typename: 'Query', countries: Array<
13   { __typename: 'Country', code: string, name: string }
14   & CountryCardCountry
15 }> };
16
17
18 export const CountryListDocument: DocumentNode = {"kind": "Document", "definitions": [{"kind": "OperationDefinit
19
20 /**
21  * __useCountryList__
22  *
23  * To run a query within a React component, call `useCountryList` and pass it any options that fit your need
24  * When your component renders, `useCountryList` returns an object from Apollo Client that contains loading,
25  * you can use to render your UI.
26  *
27  * @param baseOptions options that will be passed into the query, supported options are listed on: 
```


Кодогенерация: плагин

```
1  const plugin: CodegenPlugin<BootstrapRawConfig>['plugin'] = (schema, documents, config) => {
2    const documentsNode = documents.reduce((nodes: DocumentNode[], documentFile) => {
3      if (documentFile.document) {
4        return [documentFile.document, ...nodes]
5      }
6      return nodes
7    }, []);
8    const allAst = concatAST(documentsNode);
9    const fragments = allAst.definitions.reduce((nodes: FragmentDefinitionNode[], definition) => {
10     if (definition.kind === Kind.FRAGMENT_DEFINITION) {
11       return [definition, ...nodes]
12     }
13     return nodes
14   }, []);
15   const allFragments: LoadedFragment[] = fragments.map(
16     (fragment) => ({
17       node: fragment,
18       name: fragment.name.value,
19       onType: fragment.typeCondition.name.value,
20       isExternal: false,
21     }),
22     ...(config.externalFragments || [])
23   )
24   const visitor = new BootstrapVisitor(schema, allFragments, config, documents);
25   const result = oldVisit(allAst, {
26     leave: visitor,
27     SelectionSet: visitor.addTypenameToDocument(),
28   })
29   const definitions = result.definitions.filter((t: unknown) => typeof t === 'string').join('\n') as unknown as string;
30   const content = [visitor.fragments, definitions].join('\n');
31
32   return {
33     prepend: visitor.getImports(),
34     content,
35   }
36 }
```

Кодогенерация: пишем плагин

```
1 export const plugin: CodegenPlugin<BootstrapConfig>['plugin'] = (schema, documents, config) => {
2   const documentsNode = documents.reduce((nodes: DocumentNode[], documentFile) => {
3     if (documentFile.document) {
4       return [documentFile.document, ...nodes]
5     }
6     return nodes
7   }, []);
8   const allAst = concatAST(documentsNode);
9   const fragments = allAst.definitions.reduce((nodes: FragmentDefinitionNode[], definition) => {
10    if (definition.kind === Kind.FRAGMENT_DEFINITION) {
11      return [definition, ...nodes]
12    }
13    return nodes
14  }, []);
15  const allFragments: LoadedFragment[] = fragments.map(
16    (fragment) => ({
17      node: fragment,
18      name: fragment.name.value,
19      onType: fragment.typeCondition.name.value,
20      isExternal: false,
21    }),
22    ...(config.externalFragments || [])
23  )
24  const visitor = new BootstrapVisitor(schema, allFragments, config, documents);
25  const result = oldVisit(allAst, {
26    // @ts-expect-error: 'leave' is not a NewVisitor['leave']
27    leave: visitor,
28    SelectionSet: visitor.addTypenameToDocument(),
29  })
30  const definitions = result.definitions.filter((t: unknown) => typeof t === 'string').join('\n') as unknown as string;
31  const content = [visitor.fragments, definitions].join('\n');
32
33  return {
34    prepend: visitor.getImports(),
35    content: content,
36  }
37 }
```

Кодогенерация: плагин

```
1 const documentsNode = documents.reduce((nodes: DocumentNode[], documentFile) => {
2   if (documentFile.document) {
3     return [documentFile.document, ...nodes]
4   }
5   return nodes
6 }, []);
7 const allAst = concatAST(documentsNode);
8 const fragments = allAst.definitions.reduce((nodes: FragmentDefinitionNode[], definition) => {
9   if (definition.kind === Kind.FRAGMENT_DEFINITION) {
10    return [definition, ...nodes]
11  }
12  return nodes
13 }, []);
14 const allFragments: LoadedFragment[] = fragments.map(
15   (fragment) => ({
16     node: fragment,
17     name: fragment.name.value,
18     onType: fragment.typeCondition.name.value,
19     isExternal: false,
20   }),
21   ...(config.externalFragments || [])
22 )
23 const visitor = new BootstrapVisitor(schema, allFragments, config, documents);
24 const result = oldVisit(allAst, {
25   leave: visitor,
26   SelectionSet: visitor.addTypenameToDocument(),
27 })
```

Кодогенерация: плагин

```
1  const documentsNode = documents.reduce((nodes: DocumentNode[], documentFile) => {
2    if (documentFile.document) {
3      return [documentFile.document, ...nodes]
4    }
5    return nodes
6  }, []);
7  const allAst = concatAST(documentsNode);
8  const fragments = allAst.definitions.reduce((nodes: FragmentDefinitionNode[], definition) => {
9    if (definition.kind === Kind.FRAGMENT_DEFINITION) {
10     return [definition, ...nodes]
11   }
12   return nodes
13 }, []);
14 const allFragments: LoadedFragment[] = fragments.map(
15   (fragment) => ({
16     node: fragment,
17     name: fragment.name.value,
18     onType: fragment.typeCondition.name.value,
19     isExternal: false,
20   }),
21   ...(config.externalFragments || [])
22 )
23 const visitor = new BootstrapVisitor(schema, allFragments, config, documents);
24 const result = oldVisit(allAst, {
25   leave: visitor,
26   SelectionSet: visitor.addTypenameToDocument(),
27 })
```

Кодогенерация: плагин

```
1  const documentsNode = documents.reduce((nodes: DocumentNode[], documentFile) => {
2    if (documentFile.document) {
3      return [documentFile.document, ...nodes]
4    }
5    return nodes
6  }, []);
7  const allAst = concatAST(documentsNode);
8  const fragments = allAst.definitions.reduce((nodes: FragmentDefinitionNode[], definition) => {
9    if (definition.kind === Kind.FRAGMENT_DEFINITION) {
10     return [definition, ...nodes]
11    }
12    return nodes
13  }, []);
14  const allFragments: LoadedFragment[] = fragments.map(
15    (fragment) => ({
16      node: fragment,
17      name: fragment.name.value,
18      onType: fragment.typeCondition.name.value,
19      isExternal: false,
20    }),
21    ...(config.externalFragments || [])
22  )
23  const visitor = new BootstrapVisitor(schema, allFragments, config, documents);
24  const result = oldVisit(allAst, {
25    leave: visitor,
26    SelectionSet: visitor.addTypenameToDocument(),
27  })
```

Кодогенерация: visitor

```
1 import {
2     LoadedFragment,
3     ClientSideBaseVisitor,
4     ClientSideBasePluginConfig,
5 } from '@graphql-codegen/visitor-plugin-common'
6
7 export class BootstrapVisitor extends ClientSideBaseVisitor<BootstrapRawConfig, BootstrapPluginConfig>
8     // code ...
9 }
```

Кодогенерация: visitor

```
1 export class BootstrapVisitor extends ClientSideBaseVisitor<BootstrapRawConfig, BootstrapPluginConfig> {
2   private fetcherFnName = 'fetcher'
3   private templates = [defaultTemplateBuild, defaultTemplateMockBuild, defaultTemplateReactQueryHook]
4
5   protected renderTemplate(props: TemplateProps) {}
6
7   protected buildOperation: BuildOperation = (
8     _node,
9     _documentVariableName,
10    _operation,
11    _operationResultType,
12    _operationVariablesTypes,
13    _hasRequiredVariables
14  ) => {
15    if (_operation === 'Query' || _operation === 'Mutation') {
16      return this.renderTemplate({
17        node: _node,
18        query: _documentVariableName,
19        operationName: _node.name?.value || 'unknown',
20        fnName: this.fetcherFnName,
21        operation: _operation.toLowerCase(),
22        operationResultType: _operationResultType,
23        variableTypes: _operationVariablesTypes,
24        hasRequiredVariables: _hasRequiredVariables,
25      })
26    }
27    return ''
28  }
29 }
```

Кодогенерация: visitor

```
1 export class BootstrapVisitor extends ClientSideBaseVisitor<BootstrapRawConfig, BootstrapPluginConfig> {
2   private fetcherFnName = 'fetcher'
3   private templates = [defaultTemplateBuild, defaultTemplateMockBuild, defaultTemplateReactQueryHook]
4
5   protected renderTemplate(props: TemplateProps) {}
6
7   protected buildOperation: BuildOperation = (
8     _node,
9     _documentVariableName,
10    _operation,
11    _operationResultType,
12    _operationVariablesTypes,
13    _hasRequiredVariables
14  ) => {
15    if (_operation === 'Query' || _operation === 'Mutation') {
16      return this.renderTemplate({
17        node: _node,
18        query: _documentVariableName,
19        operationName: _node.name?.value || 'unknown',
20        fnName: this.fetcherFnName,
21        operation: _operation.toLowerCase(),
22        operationResultType: _operationResultType,
23        variableTypes: _operationVariablesTypes,
24        hasRequiredVariables: _hasRequiredVariables,
25      })
26    }
27    return ''
28  }
29 }
```


Кодогенерация: visitor

```
1 export class BootstrapVisitor extends ClientSideBaseVisitor<BootstrapRawConfig, BootstrapPluginConfig> {
2   private fetcherFnName = 'fetcher'
3   private templates = [defaultTemplateBuild, defaultTemplateMockBuild, defaultTemplateReactQueryHook]
4
5   protected renderTemplate(props: TemplateProps) {}
6
7   protected buildOperation: BuildOperation = (
8     _node,
9     _documentVariableName,
10    _operation,
11    _operationResultType,
12    _operationVariablesTypes,
13    _hasRequiredVariables
14  ) => {
15    if (_operation === 'Query' || _operation === 'Mutation') {
16      return this.renderTemplate({
17        node: _node,
18        query: _documentVariableName,
19        operationName: _node.name?.value || 'unknown',
20        fnName: this.fetcherFnName,
21        operation: _operation.toLowerCase(),
22        operationResultType: _operationResultType,
23        variableTypes: _operationVariablesTypes,
24        hasRequiredVariables: _hasRequiredVariables,
25      })
26    }
27    return ''
28  }
29 }
```

Кодогенерация: visitor

```
1 export class BootstrapVisitor extends ClientSideBaseVisitor<BootstrapRawConfig, BootstrapPluginConfig> {
2   private fetcherFnName = 'fetcher'
3   private templates = [defaultTemplateBuild, defaultTemplateMockBuild, defaultTemplateReactQueryHook]
4
5   protected renderTemplate(props: TemplateProps) {}
6
7   protected buildOperation: BuildOperation = (
8     _node,
9     _documentVariableName,
10    _operation,
11    _operationResultType,
12    _operationVariablesTypes,
13    _hasRequiredVariables
14  ) => {
15    if (_operation === 'Query' || _operation === 'Mutation') {
16      return this.renderTemplate({
17        node: _node,
18        query: _documentVariableName,
19        operationName: _node.name?.value || 'unknown',
20        fnName: this.fetcherFnName,
21        operation: _operation.toLowerCase(),
22        operationResultType: _operationResultType,
23        variableTypes: _operationVariablesTypes,
24        hasRequiredVariables: _hasRequiredVariables,
25      })
26    }
27    return ''
28  }
29 }
```

Кодогенерация: интерфейсы к Template Builder

```
1 import type { OperationDefinitionNode } from "graphql";
2
3 export interface TemplateProps {
4     node: OperationDefinitionNode,
5     query: string,
6     fnName: string;
7     /**
8      * @value 'query' | 'mutation' | 'subscription'
9      */
10    operation: string;
11    operationResultType: string,
12    variableTypes: string,
13    hasRequiredVariables: boolean,
14 }
15
16 export interface TemplateFn {
17     (props: TemplateProps): string
18 }
```

Кодогенерация: шаблоны

```
1  const required = `  
2    fetch{{operationResultType}} =  
3      (v: {{variableTypes}}) =>  
4        {{fnName}}.{{operation}}<{{operationResultType}}>  
5          ({{query}}, v);  
6  `;  
7  
8  const partial = `  
9    fetch{{operationResultType}} =  
10     (v?: {{variableTypes}}) =>  
11       {{fnName}}.{{operation}}<{{operationResultType}}>  
12         ({{query}}, v ?? {});  
13  `;  
14  
15  const standard = `  
16    fetch{{operationResultType}} =  
17      () =>  
18        {{fnName}}.{{operation}}<{{operationResultType}}>({{query}});  
19  `;
```

Кодогенерация: шаблоны

```
1  const t: TemplateFn = (props) => {
2    const r = Mustache.render
3
4    if (props.hasRequiredVariables) {
5      return r(`export const ${required}`, props)
6    }
7
8    if (props.variableTypes && props.node.variableDefinitions?.length) {
9      return r(`export const ${partial}`, props)
10   }
11
12   return r(`export const ${standard}`, props)
13 }
14
15 export default t
```

Кодогенерация: шаблон fetch функции

```
1 import type { TemplateFn } from './types.js'
2 import Mustache from 'mustache'
3
4 const required = `
5   fetch{{operationResultType}} =
6     (v: {{variableTypes}}) =>
7       {{fnName}}.{{operation}}<{{operationResultType}}>
8         ({{query}}, v);
9 `
10 const partial = `
11   fetch{{operationResultType}} =
12     (v?: {{variableTypes}}) =>
13       {{fnName}}.{{operation}}<{{operationResultType}}>
14         ({{query}}, v ?? {});
15 `
16 const standard = `
17   fetch{{operationResultType}} =
18     () =>
19       {{fnName}}.{{operation}}<{{operationResultType}}>({{query}});
20 `
21
22 const t: TemplateFn = (props) => {
23   const r = Mustache.render
24   if (props.hasRequiredVariables) {
25     return r(`export const ${required}`, props)
26   }
27   if (props.variableTypes && props.node.variableDefinitions?.length) {
28     return r(`export const ${partial}`, props)
29   }
30   return r(`export const ${standard}`, props)
31 }
32
33 export default t
```

Кодогенерация: near-operation-file-preset

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

Кодогенерация: настраиваем кодогенерацию файлов

```
1 schema: ".codegen/schema.graphql"
2 documents:
3   - "./src/**/*.graphql"
4 ignoreNoDocuments: true
5 extensions:
6   codegen:
7     errorsOnly: true
8     generates:
9       ./types/schema.ts:
10        config:
11          defaultScalarType: unknown
12          scalars:
13            ID: "number"
14        plugins:
15          - typescript
16          - add:
17            content: "/* eslint-disable */\n\n /* Auto generated file */ \n "
```


Кодогенерация: настраиваем кодогенерацию файлов

```
1 schema: ".codegen/schema.graphql"
2 documents:
3   - "./src/**/*.graphql"
4 ignoreNoDocuments: true
5 extensions:
6   codegen:
7     errorsOnly: true
8     generates:
9       ./types/schema.ts:
10        config:
11          defaultScalarType: unknown
12          scalars:
13            ID: "number"
14        plugins:
15          - typescript
16          - add:
17            content: "/* eslint-disable */\n\n /* Auto generated file */ \n "
18      ./src:
19        preset: near-operation-file
20        presetConfig:
21          baseTypesPath: ../types/schema.ts
22          folder: _generated_
23          extension: .generated.ts
24        config:
25          skipTypename: true
26          nonOptionalTypename: true
27          preResolveTypes: false
28          namingConvention:
29            typeNames: change-case-all#pascalCase
30        plugins:
31          - typescript-operations
32          - './scripts/bootstrap-generate/dist/index.js':
33            baseFetch:
34              fetcher: './client/api.ts'
```

Кодогенерация: подготовим запросы

```
first.query.graphql M x
src > app > first.query.graphql > characterCard
1 fragment characterCard on Character {
2   id
3   name
4 }
5
6 Execute Query
7 query First {
8   characters {
9     results {
10      ...characterCard
11    }
12 }

first2.query.graphql M x
src > app > first2.query.graphql > First2
Execute Query
1 query First2($page: Int = 1) {
2   episodes(page: $page) {
3     results {
4       name
5     }
6     info {
7       next
8     }
9   }
10 }
11

first3.query.graphql M x
src > app > first3.query.graphql > First3
Execute Query
1 query First3($filterName: String!) {
2   characters(filter: { name: $filterName }) {
3     results {
4       name
5       status
6     }
7   }
8 }
9
```

Кодогенерация: запускаем генерацию файлов

Кодогенерация: запускаем генерацию файлов

```
src > app > _generated_ > first.query.generated.ts > TypeScript > CharacterCardFragment
1 import * as Types from '../..../types/schema'
2
3 import { fetcher as fetcher } from '../..../client/api'
4 import gql from 'graphql-tag' 2.2k (gzipped: 1.1k)
5
6 export type CharacterCardFragment = { __typename: 'Character' } & Pick<
7   ...Types.Character,
8   ...'id' | 'name'
9 >
10
11 export type FirstQueryVariables = Types.Exact<{ [key: string]: never }>
12
13 export type FirstQuery = { __typename: 'Query' } & {
14   characters?: Types.Maybe<
15     { __typename: 'Characters' } & {
16       results?: Types.Maybe<
17         Array<
18           Types.Maybe<
19             { __typename: 'Character' } & Pick<
20               Types.Character,
21               'id' | 'name'
22             >
23           >
24         >
25       >
26     }
27   >
28 }
29
30 export const CharacterCardFragmentDoc = gql`
31   fragment characterCard on Character {
32     id
33     name
34 }
```

```
simple-graphql-bootstrap on main [x!+?] via v18.1
8.2
• → yarn graphql:generate
yarn run v1.22.21
$ graphql-codegen-esm ./graphqlrc.yml
✓ Parse Configuration
✓ Generate outputs
✨ Done in 2.08s.

simple-graphql-bootstrap on main [x!+?] via v18.1
8.2 took 2s
○ +
```

Кодогенерация: запускаем генерацию файлов

The image shows a VS Code editor window with the following content:

EXPLORER: SIMPLE-GRAPHQL-BOOTSTRAP

- .codegen
- .history
- .vscode
- client
- dist
- scripts
- src/app
 - _generated_
 - charterFields.frag... U
 - first.query.gen... 3, U
 - first2.query.gener... U
 - first3.query.gener... U
 - charterFields.frag... U
 - first.query.graphql M
 - first2.query.graphql M
 - first3.query.graphql M
 - loadData.test.ts M
 - loadData.ts M
 - loadDataDemo.ts U
 - notEmpty.ts U
- tests
- types
- .env U
- .eslintrc.json A
- .gitignore A
- .graphqlrc.yml M
- .nvmrc A
- .prettierrc.js M

Terminal: src > app > _generated_ > first.query.generated.ts > ...

```
30 nst CharacterCardFragmentDoc = gql`
31   ... characterCard on Character {
32     id
33     name
34   }
35 `
36 nst FirstDocument = gql`
37   First {
38     characters {
39       results {
40         ... characterCard
41         __typename
42       }
43       __typename
44     }
45     __typename
46   }
47   characterCardFragmentDoc}
48 `
49 nst fetchFirstQuery = () => fetcher.query<FirstQuery>(FirstDocument)
50 `
```

Output Console:

```
simple-graphql-bootstrap on main [x!+?] via v18.1 8.2
• → yarn graphql:generate
yarn run v1.22.21
$ graphql-codegen-esm ./graphqlrc.yml
✓ Parse Configuration
✓ Generate outputs
Done in 2.08s.

simple-graphql-bootstrap on main [x!+?] via v18.1 8.2 took 2s
```

Кодогенерация: магия

Кодогенерация: магия

```
first.query.graphql M | loadDataDemo.ts, U • | zsh + v | ... x
```

```
src > app > loadDataDemo.ts > ...
```

```
1 import {
2   fetchFirstQuery,
3   CharacterCardFragment,
4 } from './_generated_/first.query.generated'
5 import { notEmpty } from './notEmpty'
6
7 const query1 = await fetchFirstQuery()
8
9 function print(characters?: CharacterCardFragment[]) {
10   if (characters) {
11     characters.forEach(({ name }) => {
12       console.log('first1', name)
13     })
14   }
15 }
16
17 const characters = |
```

```
simple-graphql-bootstrap on main [x!+?] via v18.1
8.2
• → yarn graphql:generate
yarn run v1.22.21
$ graphql-codegen-esm ./graphqlrc.yml
✓ Parse Configuration
✓ Generate outputs
✦ Done in 2.08s.

simple-graphql-bootstrap on main [x!+?] via v18.1
8.2 took 2s
○ → npx ts-node src/app/loadDataDemo.ts |
```

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: __typenameе 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  })
```

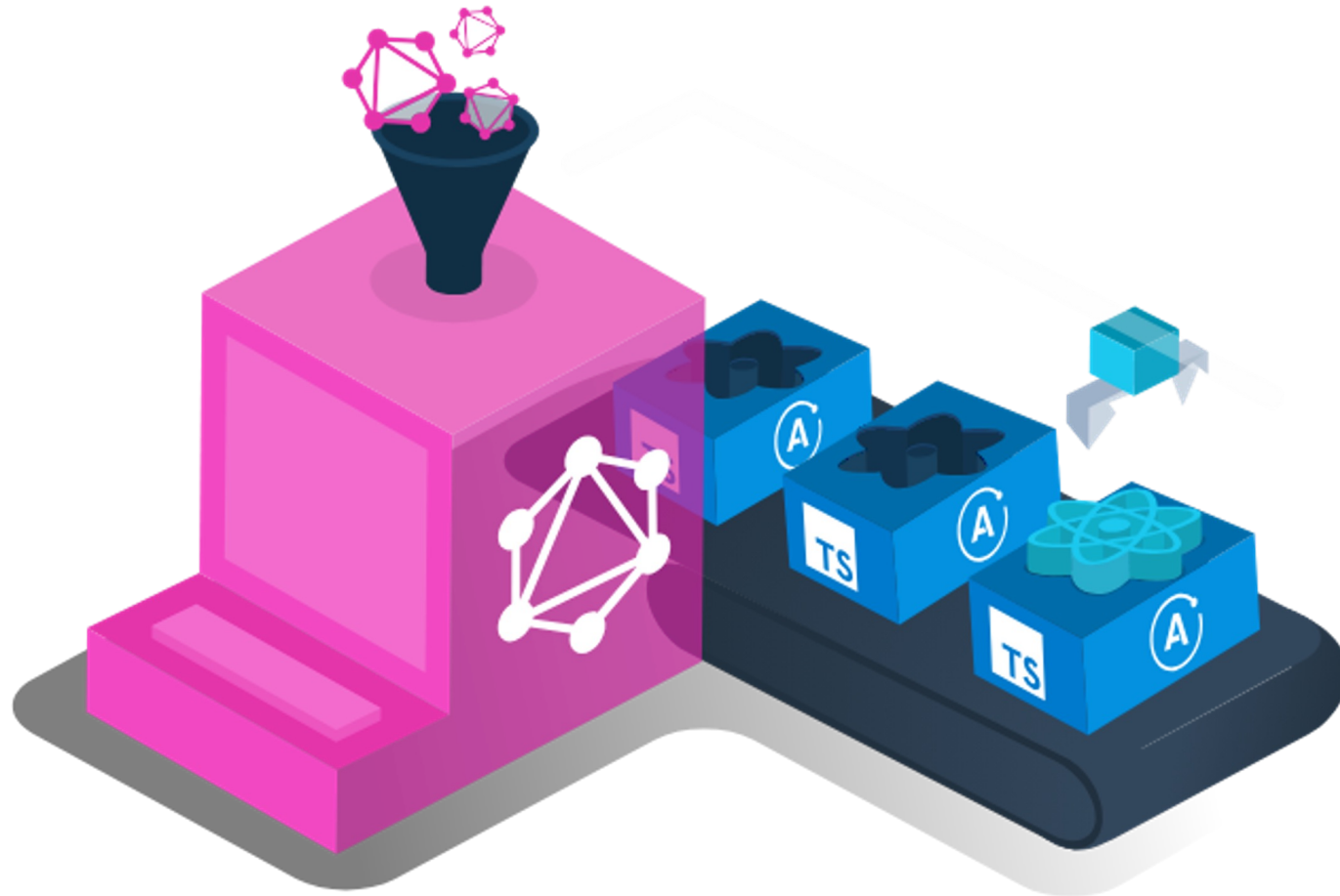
Кодогенерация: __typenameе ключ к успеху

```
1 export const CharacterCardFragmentDoc = gql`
2   fragment characterCard on Character {
3     id
4     name
5   }
6 `
7 export const FirstDocument = gql`
8   query First {
9     characters {
10      results {
11        ...characterCard
12        __typename
13      }
14      __typename
15    }
16    __typename
17  }
18  ${CharacterCardFragmentDoc}
19 `
```

Кодогенерация: __typenameе ключ к успеху

```
90
91 const renderEventItem = (event: SystemNotificationEvent) => {
92   switch (event.__typename) {
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



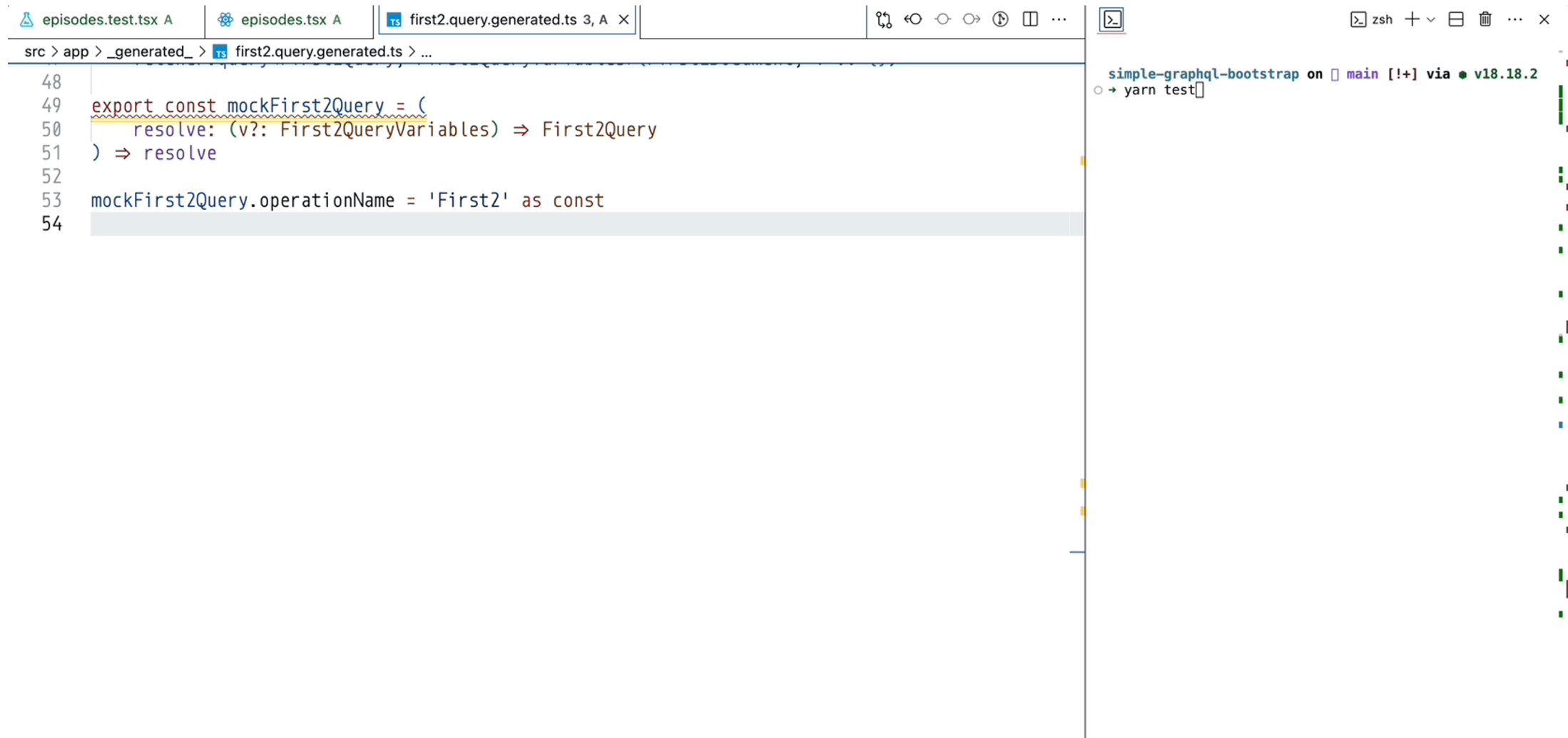
Тесты: генерация мока для операции

```
1 import type { TemplateFn } from './types.js';
2 import Mustache from 'mustache';
3
4 const r = Mustache.render;
5
6 const required = `
7   mock{{operationResultType}} =
8     (mockFn: (v: {{variableTypes}}) => {{operationResultType}}) => mockFn;
9 `;
10
11 const partial = `
12   mock{{operationResultType}} =
13     (mockFn: (v?: {{variableTypes}}) => {{operationResultType}}) => mockFn;
14 `;
15
16 const standard = `
17   mock{{operationResultType}} =
18     (mockFn: () => {{operationResultType}}) => mockFn;
19 `;
20
21
22 const t: TemplateFn = (props) => {
23   if (props.hasRequiredVariables) {
24     return r(`export const ${required}`, props);
25   }
26
27   if (props.variableTypes && props.node.variableDefinitions?.length) {
28     return r(`export const ${partial}`, props);
29   }
30
31   return r(`export const ${standard}`, props);
32 }
33
34 export default t;
```

Тесты: генерация мока для операции

```
1 import type { TemplateFn } from './types.js';
2 import Mustache from 'mustache';
3
4 const r = Mustache.render;
5
6 const required = `
7   mock{{operationResultType}} =
8     (mockFn: (v: {{variableTypes}}) => {{operationResultType}}) => mockFn;
9 `;
10
11 const partial = `
12   mock{{operationResultType}} =
13     (mockFn: (v?: {{variableTypes}}) => {{operationResultType}}) => mockFn;
14 `;
15
16 const standard = `
17   mock{{operationResultType}} =
18     (mockFn: () => {{operationResultType}}) => mockFn;
19 `;
20
21
22 const t: TemplateFn = (props) => {
23   if (props.hasRequiredVariables) {
24     return r(`export const ${required}`, props);
25   }
26
27   if (props.variableTypes && props.node.variableDefinitions?.length) {
28     return r(`export const ${partial}`, props);
29   }
30
31   return r(`export const ${standard}`, props);
32 }
33
34 export default t;
```

Тесты: поехали



The image shows a code editor with three tabs: 'episodes.test.tsx A', 'episodes.tsx A', and 'first2.query.generated.ts 3, A X'. The active tab is 'first2.query.generated.ts 3, A X'. The code in the editor is as follows:

```
48  
49 export const mockFirst2Query = (  
50   resolve: (v?: First2QueryVariables) => First2Query  
51 ) => resolve  
52  
53 mockFirst2Query.operationName = 'First2' as const  
54
```

Below the code editor is a terminal window. The terminal shows the following output:

```
simple-graphql-bootstrap on main [!+] via v18.18.2  
o → yarn test
```

Тесты: мокаем сеть

```
1 import { Agent, MockAgent, setGlobalDispatcher } from 'undici';
2 import { fetcher } from '../client/api'
3
4 export const createMockRequest = (
5   operationName: string,
6   disableNetConnect = false
7 ) => {
8   const mockAgent = new MockAgent({ connections: 1 })
9   const client = mockAgent.get(fetcher.origin)
10  const request = client.intercept({
11    method: 'POST',
12    path: '/graphql',
13    query: {
14      operationName,
15    },
16  })
17  const clear = async () => {
18    setGlobalDispatcher(new Agent())
19    await client.close()
20  }
21
22  if (disableNetConnect) {
23    mockAgent.disableNetConnect()
24  }
25
26  setGlobalDispatcher(mockAgent)
27
28  return { request, clear }
29 }
```

Тесты: мокаем сеть

```
1 import { Agent, MockAgent, setGlobalDispatcher } from 'undici';
2 import { fetcher } from '../client/api'
3
4 export const createMockRequest = (
5   operationName: string,
6   disableNetConnect = false
7 ) => {
8   const mockAgent = new MockAgent({ connections: 1 })
9   const client = mockAgent.get(fetcher.origin)
10  const request = client.intercept({
11    method: 'POST',
12    path: '/graphql',
13    query: {
14      operationName,
15    },
16  })
17  const clear = async () => {
18    setGlobalDispatcher(new Agent())
19    await client.close()
20  }
21
22  if (disableNetConnect) {
23    mockAgent.disableNetConnect()
24  }
25
26  setGlobalDispatcher(mockAgent)
27
28  return { request, clear }
29 }
```

Тесты: мокаем сеть

```
1 import { Agent, MockAgent, setGlobalDispatcher } from 'undici';
2 import { fetcher } from '../client/api'
3
4 export const createMockRequest = (
5   operationName: string,
6   disableNetConnect = false
7 ) => {
8   const mockAgent = new MockAgent({ connections: 1 })
9   const client = mockAgent.get(fetcher.origin)
10  const request = client.intercept({
11    method: 'POST',
12    path: '/graphql',
13    query: {
14      operationName,
15    },
16  })
17  const clear = async () => {
18    setGlobalDispatcher(new Agent())
19    await client.close()
20  }
21
22  if (disableNetConnect) {
23    mockAgent.disableNetConnect()
24  }
25
26  setGlobalDispatcher(mockAgent)
27
28  return { request, clear }
29 }
```



Тесты: мок GraphQL операций

```
1 import { createMockRequest } from './createMockRequest'
2
3 const mockGraphQL = (operationName: string, disableNetConnect = true) => {
4   const { request, clear } = createMockRequest(
5     operationName,
6     disableNetConnect
7   )
8
9   return {
10    reply<T extends (...args: any[]) => object>(
11      resolver: ResolveFn<T>
12    ) {
13      request.reply(function replyHandler(props) {
14        const body = typeof props.body === 'string'
15          ? JSON.parse(props.body)
16          : undefined
17
18        return {
19          statusCode: 200,
20          data: { data: resolver(body?.variables ?? undefined) },
21        }
22      })
23    },
24    replyWithError: <S extends number, E extends object>(
25      status: S,
26      error: E
27    ) => request.reply(status, error),
28    clear,
29  }
30 }
```

Тесты: мок GraphQL операций

```
1 import { createMockRequest } from './createMockRequest'
2
3 const mockGraphQL = (operationName: string, disableNetConnect = true) => {
4   const { request, clear } = createMockRequest(
5     operationName,
6     disableNetConnect
7   )
8
9   return {
10    reply<T extends (...args: any[]) => object>(
11      resolver: ResolveFn<T>
12    ) {
13      request.reply(function replyHandler(props) {
14        const body = typeof props.body === 'string'
15          ? JSON.parse(props.body)
16            : undefined
17
18        return {
19          statusCode: 200,
20          data: { data: resolver(body?.variables ?? undefined) },
21        }
22      })
23    },
24    replyWithError: <S extends number, E extends object>(
25      status: S,
26      error: E
27    ) => request.reply(status, error),
28    clear,
29  }
30 }
```


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: 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

```
episodes.test.tsx A X | episodes.tsx A | first2.query.generated.ts 3, A | zsh + v | ... X
tests > __tests__ > episodes.test.tsx > describe('<Episodes/>') callback > test('correct render with click button show episodes') callback > result
9 describe('<Episodes/>', () => {
10   test('correct render with click button show episodes', async () => {
11     const expectedHTML = '<ul><li>Test-1</li></ul>';
12     const result = render(<Episodes />)
13     const mock = mockGraphQL(mockFirst2Query.operationName)
14     mock.reply(
15       mockFirst2Query(() => ({
16         __typename: 'Query',
17         episodes: {
18           __typename: 'Episodes',
19           results: [
20             {
21               __typename: 'Episode',
22               name: 'Test-1',
23             },
24           ],
25         },
26       })),
27     )
28
29     if (result.getByRole('button')) {
30       const jumpNextMicroTask = new Promise((r) => setTimeout(r, 1))
31       fireEvent(
32         result.getByRole('button'),
33         new MouseEvent('click', {
34           bubbles: true,
35         })
36       )
37       await jumpNextMicroTask

```

```
simple-graphql-bootstrap on main [!+] via v18.18.2
o → yarn test
```



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



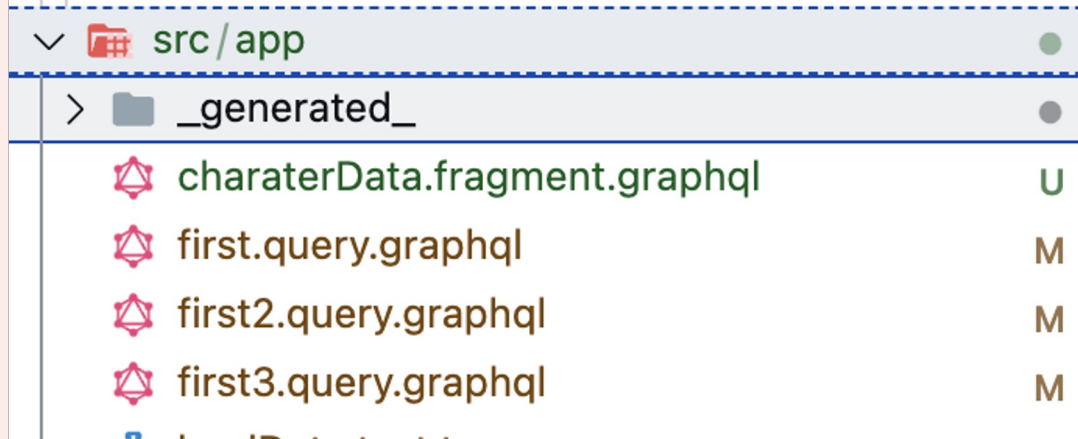
Операции и фрагменты только в файлах

src/app	
> _generated_	
charaterData.fragment.graphql	U
first.query.graphql	M
first2.query.graphql	M
first3.query.graphql	M

Фиксируем договоренности



Операции и фрагменты только в файлах

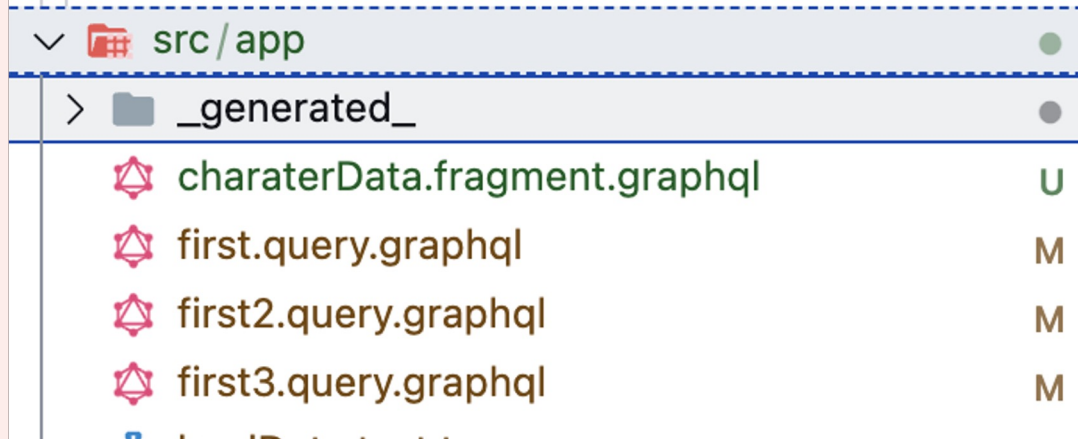


Не пишем запросы во вью слое, не собираем `template literals` запросы

Фиксируем договоренности



Операции и фрагменты только в файлах



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

```
1 function App() {
2   const allFilmsWithVariablesQueryDocument = graphql(/* GraphQL */ `
3     query allFilmsWithVariablesQuery($first: Int!) {
4       allFilms(first: $first) {
5         edges {
6           node {
7             id
8             title
9           }
10        }
11      }
12    }
13  `)
14  const { data } = useQuery({
15    queryKey: ['films'],
16    queryFn: async () =>
17      request(
18        'https://swapi-graphql.netlify.app/.netlify/functions/index',
19        allFilmsWithVariablesQueryDocument,
20        { first: 10 },
21      ),
22  })
23  // ...
24 }
```

Eslint

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


Rules

```
1 "prettier/prettier": [2, { "parser": "graphql" }],
2 "spaced-comment": "off",
3 "@graphql-eslint/avoid-duplicate-fields": "error",
4 "@graphql-eslint/executable-definitions": "error",
5 "@graphql-eslint/fields-on-correct-type": "error",
6 "@graphql-eslint/fragments-on-composite-type": "error",
7 "@graphql-eslint/known-argument-names": "error",
8 "@graphql-eslint/known-directives": "error",
9 "@graphql-eslint/known-type-names": "error",
10 "@graphql-eslint/no-anonymous-operations": "error",
11 "@graphql-eslint/no-deprecated": "error",
12 "@graphql-eslint/no-unused-variables": "error",
13 "@graphql-eslint/provided-required-arguments": "error",
14 "@graphql-eslint/scalar-leafs": "error",
15 "@graphql-eslint/unique-argument-names": "error",
16 "@graphql-eslint/unique-input-field-names": "error",
17 "@graphql-eslint/unique-variable-names": "error",
18 "@graphql-eslint/value-literals-of-correct-type": "error",
19 "@graphql-eslint/variables-are-input-types": "error",
20 "@graphql-eslint/variables-in-allowed-position": "error",
21 "@graphql-eslint/require-id-when-available": "off",
```

Rules

```
1 "prettier/prettier": [2, { "parser": "graphql" }],
2 "spaced-comment": "off",
3 "@graphql-eslint/avoid-duplicate-fields": "error",
4 "@graphql-eslint/executable-definitions": "error",
5 "@graphql-eslint/fields-on-correct-type": "error",
6 "@graphql-eslint/fragments-on-composite-type": "error",
7 "@graphql-eslint/known-argument-names": "error",
8 "@graphql-eslint/known-directives": "error",
9 "@graphql-eslint/known-type-names": "error",
10 "@graphql-eslint/no-anonymous-operations": "error",
11 "@graphql-eslint/no-deprecated": "error",
12 "@graphql-eslint/no-unused-variables": "error",
13 "@graphql-eslint/provided-required-arguments": "error",
14 "@graphql-eslint/scalar-leafs": "error",
15 "@graphql-eslint/unique-argument-names": "error",
16 "@graphql-eslint/unique-input-field-names": "error",
17 "@graphql-eslint/unique-variable-names": "error",
18 "@graphql-eslint/value-literals-of-correct-type": "error",
19 "@graphql-eslint/variables-are-input-types": "error",
20 "@graphql-eslint/variables-in-allowed-position": "error",
21 "@graphql-eslint/require-id-when-available": "off",
```

Rules

```
1 // 20 is backend constant value
2 "@graphql-eslint/selection-set-depth": ["error", { "maxDepth": 20 }],
3 "@graphql-eslint/naming-convention": [
4   "error",
5   {
6     "VariableDefinition": "camelCase",
7     "OperationDefinition": {
8       "style": "PascalCase",
9       "forbiddenPrefixes": ["Query", "Mutation", "Subscription", "Get"],
10      "forbiddenSuffixes": ["Query", "Mutation", "Subscription"]
11    },
12    "FragmentDefinition": {
13      "style": "camelCase",
14      "forbiddenPrefixes": ["Fragment"],
15      "forbiddenSuffixes": ["Fragment"]
16    }
17  }
18 ]
```

Rules

```
1  {
2    "@graphql-eslint/match-document-filename": [
3      "error",
4      {
5        "fileExtension": ".graphql",
6        "query": { "style": "camelCase", "suffix": ".query" },
7        "mutation": { "style": "camelCase", "suffix": ".mutation" },
8        "fragment": { "style": "camelCase", "suffix": ".fragment" }
9      }
10   ]
11 }
```

Rules

```
1 {
2   "@graphql-eslint/match-document-filename": [
3     "error",
4     {
5       "fileExtension": ".graphql",
6       "query": { "style": "camelCase", "suffix": ".query" },
7       "mutation": { "style": "camelCase", "suffix": ".mutation" },
8       "fragment": { "style": "camelCase", "suffix": ".fragment" }
9     }
10  ]
11 }
```

**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

```
1 import axios from 'axios'
2 import { type ASTNode, type OperationDefinitionNode } from 'graphql'
3 import type { RequestFn } from './types'
4
5 const isOperation = (node: ASTNode): node is OperationDefinitionNode => node.kind === 'OperationDefinition'
6
7 const getOperationName = (node: ASTNode): string => isOperation(node) ? node.name!.value : 'unknown'
8
9 export const request:<U extends URL>(url: U) => RequestFn = (url) => async (query, variables) => {
10   const isDocumentNode = typeof query !== 'string';
11   const operationName = isDocumentNode ? getOperationName(query.definitions[0]) : 'unknown';
12   operationName && url.searchParams.append('operationName', operationName)
13   +   const { data } = await axios.request({
14   +     url: `${url}`,
15   +     method: 'post',
16   +     headers: {
17   +       'Content-Type': 'application/json',
18   +     },
19   +     data: {
20   +       query: isDocumentNode ? query.loc?.source.body : query,
21   +       variables: variables ?? null,
22   +       operationName,
23   +     }
24   +   })
25   return data
26 }
```

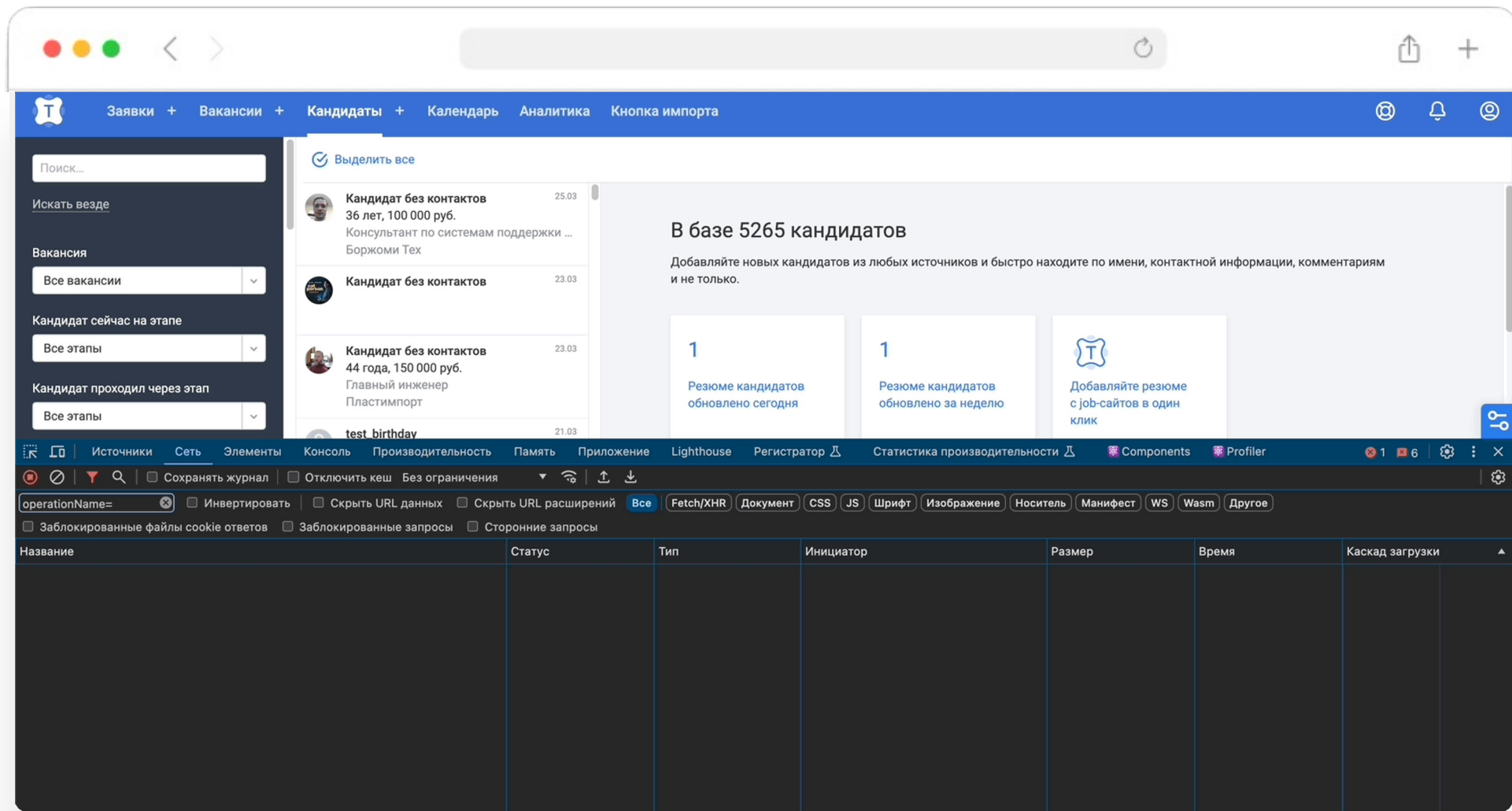
Старый fetcher

```
personsMap.js
Users > i.gorskiy > projects > tms > tms-static > app > actions > ats > personsMap > personsMap.js
108  addReaction(
171  )
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!**