

# Middlewares are awesome

Мостовой Никита,  
HeadHunter



**Frontend  
Conf**

Профессиональная  
конференция  
фронтенд-разработчиков





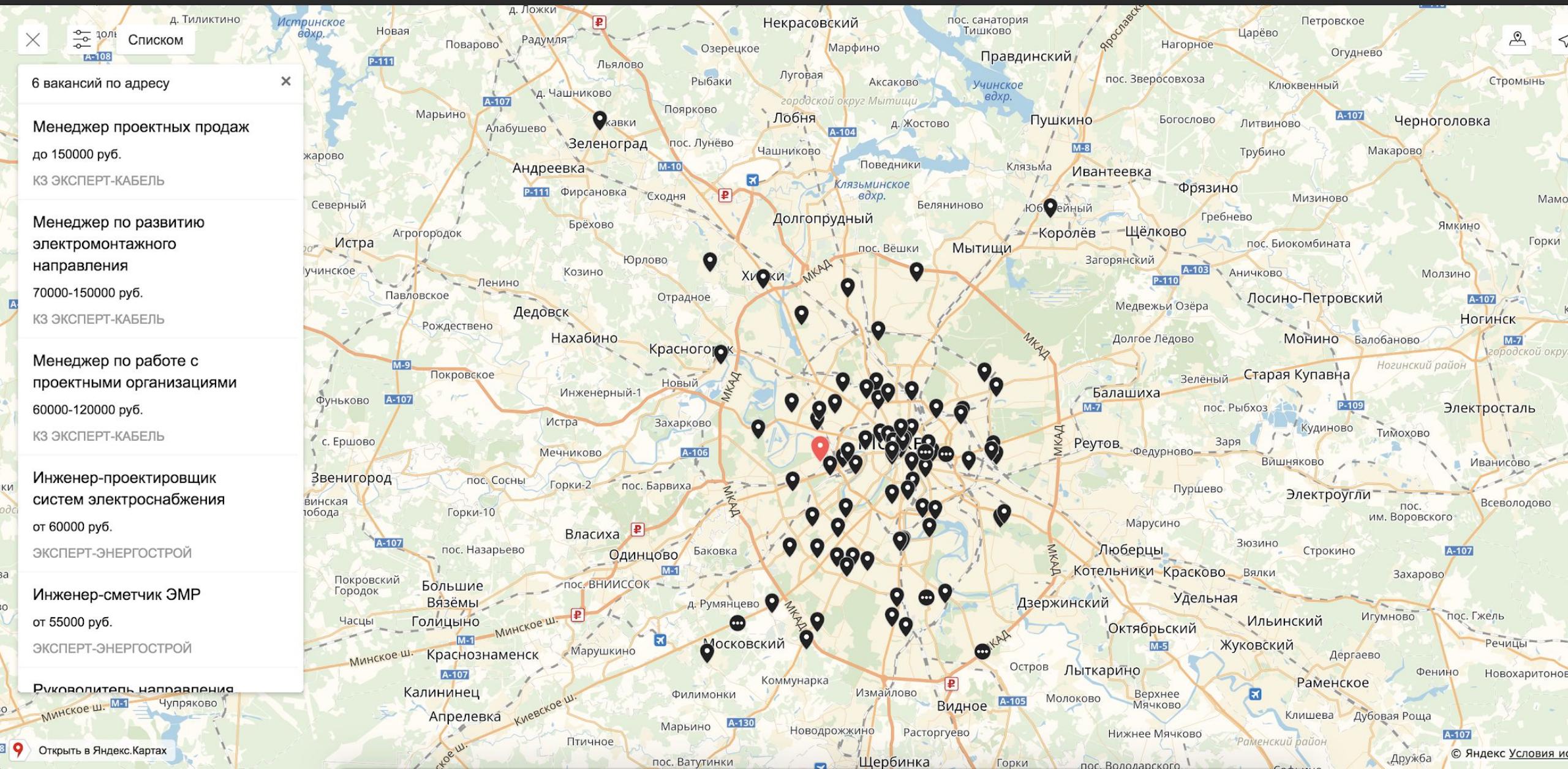
# О спикере

## Мостовой Никита

Lead Frontend Developer at  
Talantix (HeadHunter)

Twitter: [@xnimorz](https://twitter.com/@xnimorz)  
<http://xnim.ru/>





6 вакансий по адресу

## Менеджер проектных продаж до 150000 руб.

## Менеджер по развитию электромонтажного направления

70000-150000 руб.

## Менеджер по работе с проектными организациями

60000-120000 руб.

Инженер-проектировщик  
систем электроснабжения  
от 60000 руб.

Инженер-сметчик ЭМР  
от 55000 руб.

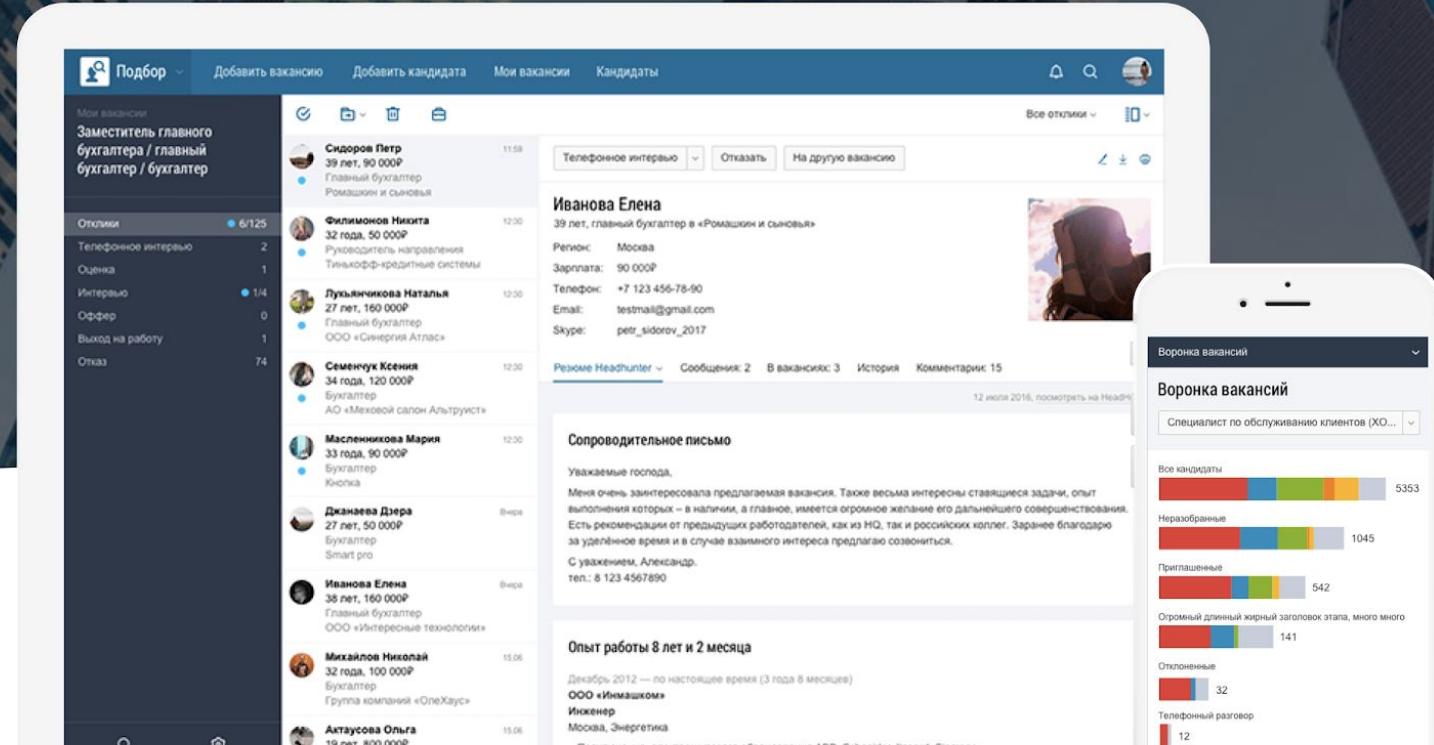
ЭКСПЕРТ-ЭНЕРГОСТРОЙ

## Вуководитель направления

 Открыть в Яндекс.Картах

# Удобная система для работы с персоналом

Когда каждый сотрудник на счету, важно экономить время на рутине. Мы собираем и автоматизируем все HR-процессы в одном удобном пространстве, у вас под рукой.

[Демодоступ](#)

The screenshot displays the Talantix software interface, which includes several recruitment modules:

- Подбор (Recruitment):** Shows a list of candidates for the position "Заместитель главного бухгалтера / главный бухгалтер / бухгалтер" (Deputy Chief Accountant / Chief Accountant / Accountant). It includes sections for Отклики (Responses), Телефонное интервью (Phone interview), Оценка (Evaluation), Интервью (Interview), Оффер (Offer), Выход на работу (Job offer), and Отказ (Decline).
- Иванова Елена (Candidate Profile):** A detailed profile for a candidate named Иванова Елена, aged 39, with a salary requirement of 90,000 RUB. It lists her experience as a chief accountant at Romashkin & Sinyavina, her phone number (+7 123 456-78-90), email (testmail@gmail.com), and Skype (petr\_sidorov\_2017). It also shows her resume HeadHunter.
- Сопроводительное письмо (Reference letter):** A template letter addressed to "Уважаемые господа," expressing interest in the advertised position and mentioning previous work experience at Smart pro.
- Опыт работы 8 лет и 2 месяца (Work experience 8 years and 2 months):** A summary of the candidate's work history from December 2012 to the present, listing various companies and roles.
- Воронка вакансий (Vacancy funnel):** A visualization showing the status of candidates across different stages: Все кандидаты (All candidates) - 5353, Неразобранные (Unread) - 1045, Приглашенные (Invited) - 542, Огромный длинный жирный заголовок этапа, много много (Large, long, bold stage header, many many) - 141, Отклоненные (Rejected) - 32, and Телефонный разговор (Telephone call) - 12.

The interface is designed to be user-friendly, with a dark theme and clear navigation. A mobile phone icon in the bottom right corner indicates the software is accessible via mobile devices.

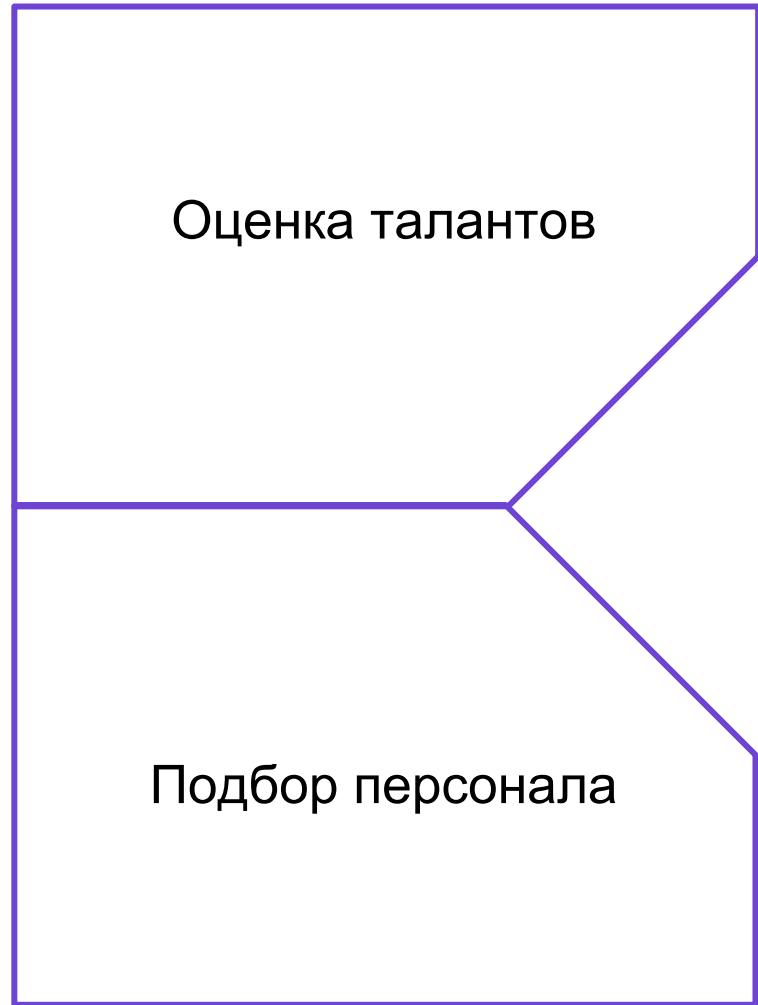


# Модульный проект

Оценка талантов

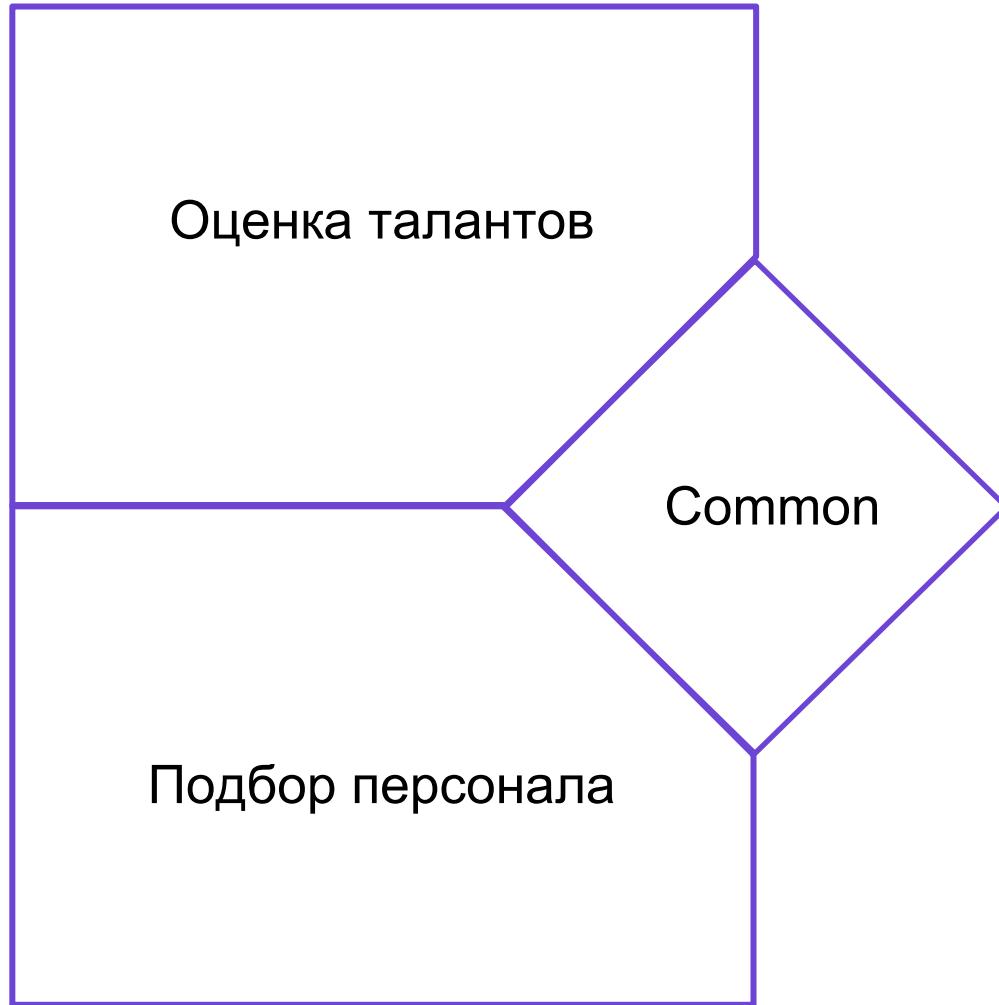


# Модульный проект



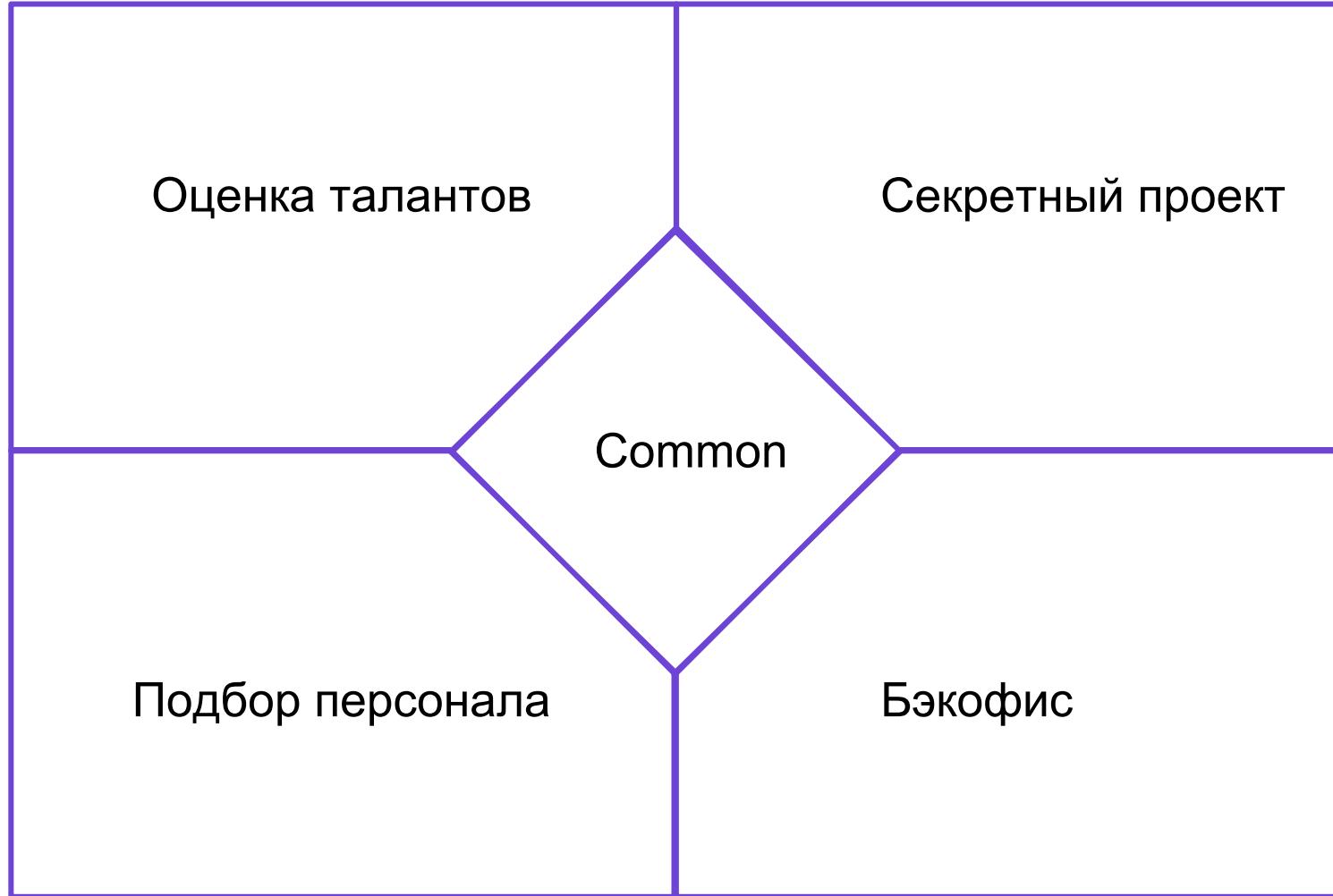


# Модульный проект





# Модульный проект



# Каким был проект?

React components view

React-redux  
connect

Business logic: action creators via redux-thunk

Actions

Data logic: reducers

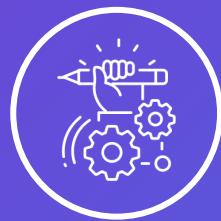
React components view

React-redux  
connect

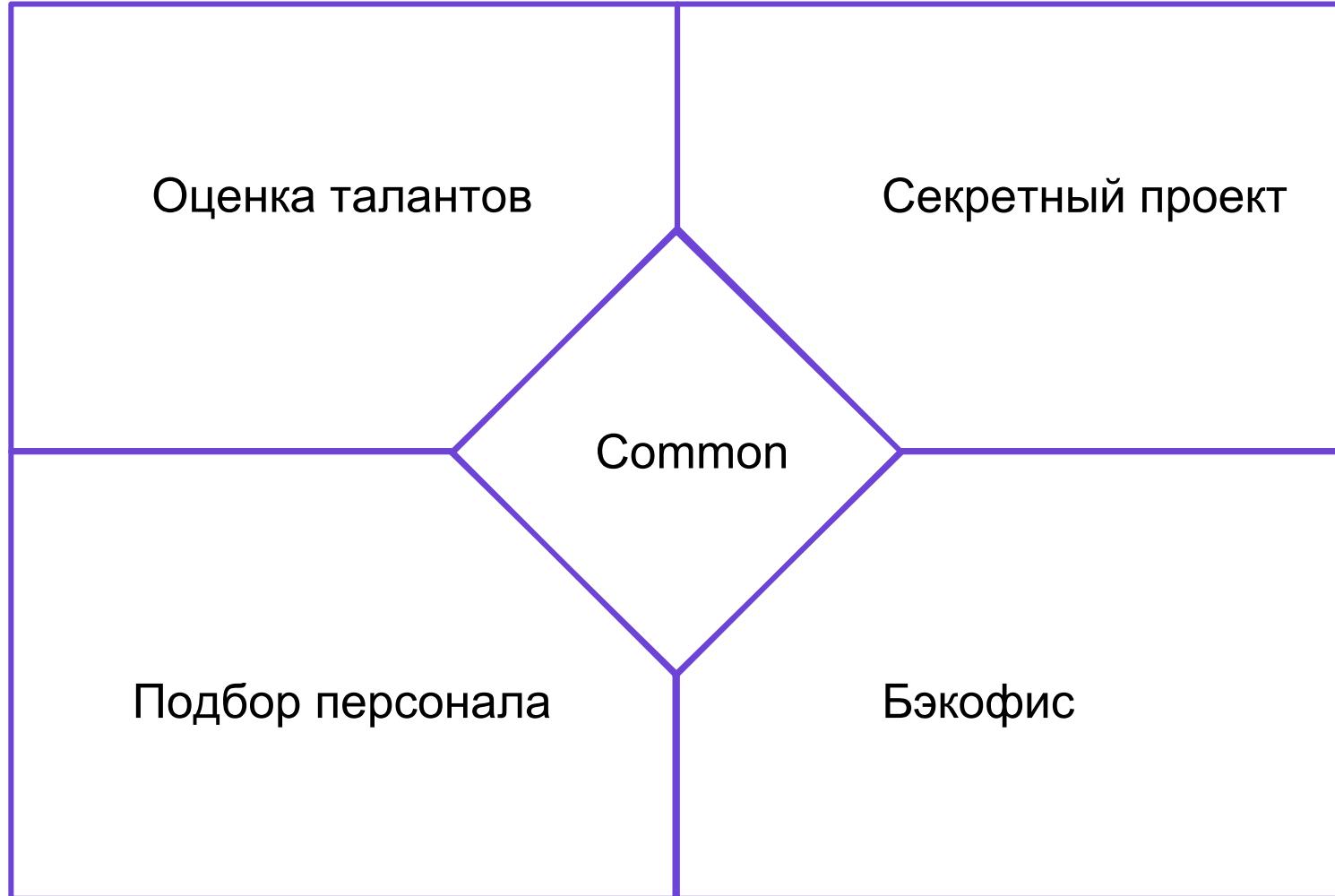
Business logic: action creators via redux-thunk

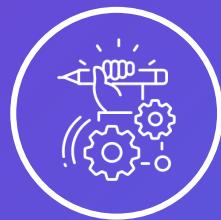
Actions

Data logic: reducers



# Модульный проект. React components

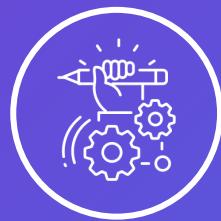




# Иерархия компонентов

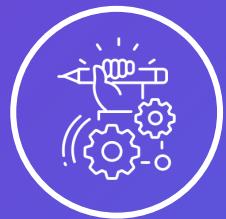
## ◀ **components**

- ▶ adminModule
- ▶ assessmentsModule
- ▶ atsModule
- ▶ common
- ▶ extensionModule



# Модульный проект. React components

UI-toolkit в common-компонентах



# Модульный проект. React components

UI-toolkit в common-компонентах

Тупые и умные компоненты



# Модульный проект. React components

UI-toolkit в common-компонентах

Тупые и умные компоненты

React-redux connect

React components view

React-redux  
connect

Business logic: action creators via redux-thunk

Actions

Data logic: reducers

React components view

React-redux  
connect

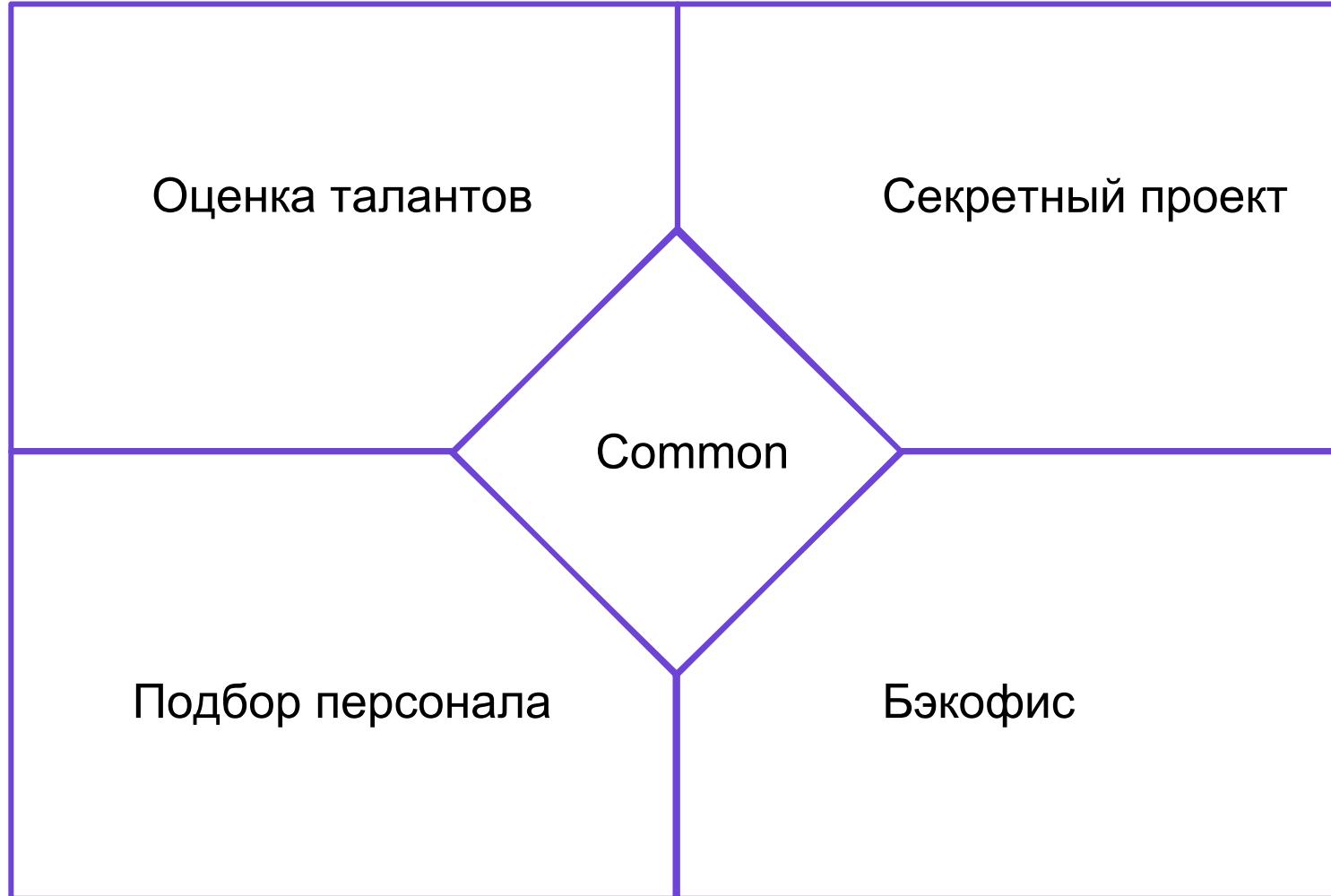
Business logic: action creators via redux-thunk

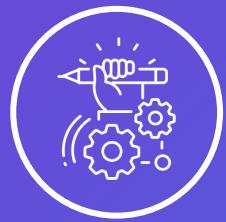
Actions

Data logic: reducers



# Модульный проект. React store





# Модульный проект. Redux store

Combine reducers



# Модульный проект. Redux store

- account: {firstName: "Никита",  
► areas: {}  
► ats: {vacancies: {...}, calendar  
► command: {state: "COMPLETE"}  
► confirm: {}  
► features: {}  
► filters: {}



# Модульный проект. Redux store

- account: {firstName: "Никита",  
► areas: {}  
► ats: {vacancies: {...}, calendar  
► command: {state: "COMPLETE"}  
► confirm: {}  
► features: {}  
► filters: {}

combineReducers



# Модульный проект. Redux store

- account: {firstName: "Никита",  
► areas: {}  
► ats: {vacancies: {...}, calendar  
► command: {state: "COMPLETE"}  
► confirm: {}  
► features: {}  
► filters: {}

combineReducers



# Модульный проект. Redux store

Combine reducers

InjectReducers — <https://habr.com/company/hh/blog/310524/>

React components view

React-redux  
connect

Business logic: action creators via redux-thunk

Actions

Data logic: reducers

React components view

React-redux  
connect

Business logic: action creators via redux-thunk

Actions

Data logic: reducers

React components view

React-redux  
connect

Business logic: action creators via redux-thunk

**Магия**

Actions

Data logic: reducers





# Чем мы сегодня займемся?

Разберем структуру middleware, его  
составные части и выполняемые задачи



# Чем мы сегодня займемся?

Разберем структуру middleware, его  
составные части и выполняемые задачи

Опишем спектр задач отлично решаемых с  
помощью middleware

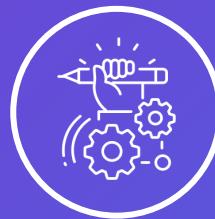


# Чем мы сегодня займемся?

Разберем структуру middleware, его составные части и выполняемые задачи

Опишем спектр задач отлично решаемых с помощью middleware

Посмотрим, как с помощью middleware можно разложить логику приложения на MVC



# Чем мы сегодня займемся?

Разберем структуру middleware, его составные части и выполняемые задачи

Опишем спектр задач отлично решаемых с помощью middleware

Посмотрим, как с помощью middleware можно разложить логику приложения на MVC

Поговорим как о положительных чертах middleware, так и о недостатках



# Первое знакомство новичка с middleware





# Первый вопрос новичков в Redux

[Questions](#)[Developer Jobs](#)[Tags](#)[Users](#)

## Handling async request with React, Redux and Axios?



15

I am new to React JS and Redux and it has been too overwhelming to get going. I am trying to make a POST request using Axios, but I am unable to make it. May be I am missing something in the container file. Below is the code. Check [plnkr](#)



**Update:** I am getting @@redux-form/SET\_SUBMIT\_SUCCEEDED message after submitting. But when I am checking in the network tab, I don't see the call to API. And also when I am consoling the submitted values, I see only name and fullname values. It doesn't consist of logo and details. What am I missing?



2



# Первый вопрос новичков в Redux

Handlin Google

async request redux

Все Видео Картинки Новости Покупки Ещё Настройки Инструменты

Результатов: примерно 173 000 (0,35 сек.)

Совет. По этому запросу вы можете найти сайты на [русском языке](#). Указать предпочтительные языки для результатов поиска можно в разделе [Настройки](#).

[React + Redux Tutorial Part III: Async Redux](#)  
www.thegreatcodeadventure.com/react-redux-tutorial-part-iii... ▾ Перевести эту страницу  
6 сент. 2016 г. - In order for our CatsPage component to display all the wonderful cats, our React app needs to make an API **request**, receive the cats payload, and somehow pass that data down to the CatsPage component. This is where our store, action creators and reducers will come in. Let's break down this flow before ...

[redux/AsyncActions.md at master · reactjs/redux · GitHub](#)  
<https://github.com/reactjs/redux/blob/.../AsyncActions.md> ▾ Перевести эту страницу  
Async Action Creators. Finally, how do we use the synchronous action creators we defined earlier together with network **requests**? The standard way to do it with **Redux** is to use the **Redux Thunk** middleware. It comes in a separate package called **redux-thunk**. We'll explain how middleware works in general later; for now, ...



# Давайте решим задачу без middleware?

Questions   Developer Jobs   Tags   Users   Search...

Handling Google

async request redux

Все   Видео   Картинки   Новости   Покупки   Ещё   Настройки   Инструменты

15

I am a F

15

coi

Up wh su am 2

Результатов: примерно 173 000 (0,35 сек.)

Совет. По этому запросу вы можете [найти сайты на русском языке](#). Указать предпочтительные языки для результатов поиска можно в разделе [Настройки](#).

[React + Redux Tutorial Part III: Async Redux](#)  
[www.thegreatcodeadventure.com/react-redux-tutorial-part-iii...](http://www.thegreatcodeadventure.com/react-redux-tutorial-part-iii...) ▾ [Перевести эту страницу](#)  
6 сент. 2016 г. - In order for our CatsPage component to display all the wonderful cats, our React app needs to make an API **request**, receive the cats payload, and somehow pass that data down to the CatsPage component. This is where our store, action creators and reducers will come in. Let's break down this flow before ...

[redux/AsyncActions.md at master · reactjs/redux · GitHub](#)  
<https://github.com/reactjs/redux/blob/.../AsyncActions.md> ▾ [Перевести эту страницу](#)  
Async Action Creators. Finally, how do we use the synchronous action creators we defined earlier together with network **requests**? The standard way to do it with **Redux** is to use the **Redux Thunk** middleware. It comes in a separate package called **redux-thunk** . We'll explain how middleware works in general later; for now, ...

36

FC 2018 Frontend Conf



# Запрос в react-компоненте, затем dispatch

```
class Test extends Component {  
  componentDidMount() {  
    fetch(API_URL).then((result) => result.json())  
      .then((result) => this.props.receiveData(result))  
  }  
  render() {  
    return <div />;  
  }  
}
```



# Запрос в react-компоненте, затем dispatch

```
class Test extends Component {  
  componentDidMount() {  
    fetch(API_URL).then((result) => result.json())  
      .then((result) => this.props.receiveData(result))  
  }  
  render() {  
    return <div />;  
  }  
}
```



# Запрос в react-компоненте, затем dispatch

Лапша — вьюха и бизнес-логика в одном месте



# Запрос в react-компоненте, затем dispatch

Лапша — вьюха и бизнес-логика в одном месте

Сложнее избегать дублирования, сложнее следить за кодом



# Передавать dispatch в action creator

```
class Test extends Component {  
  componentDidMount() {  
    fetchUser(this.props.id, this.props.dispatch);  
  }  
}  
  
function fetchUser(id, dispatch) {  
  fetch(API_URL)  
    .then((result) => result.json())  
    .then((result) => dispatch({type: "RECEIVE_USER", payload: result}))  
}
```



# Передавать dispatch в action creator

```
class Test extends Component {  
  componentDidMount() {  
    fetchUser(this.props.id, this.props.dispatch);  
  }  
}  
  
function fetchUser(id, dispatch) {  
  fetch(API_URL)  
    .then((result) => result.json())  
    .then((result) => dispatch({type: "RECEIVE_USER", payload: result}))  
}
```



# Но вернемся к нашему новичку





# Но вернемся к нашему новичку...

redux-thunk

redux-saga

redux-logic



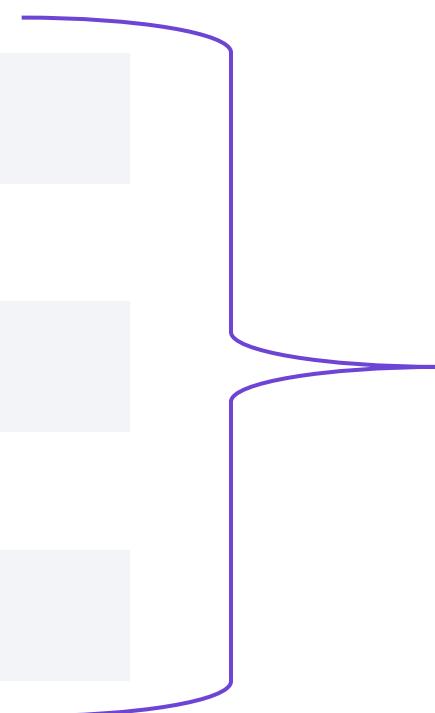
# Но вернемся к нашему новичку...

redux-thunk

redux-saga

redux-logic

Middleware





# Redux-thunk — простейший middleware

```
function createThunkMiddleware(extraArgument) {  
  return ({ dispatch, getState }) => next => action => {  
    if (typeof action === 'function') {  
      return action(dispatch, getState, extraArgument);  
    }  
    return next(action);  
  };  
}  
  
const thunk = createThunkMiddleware();  
thunk.withExtraArgument = createThunkMiddleware;  
export default thunk;
```



# Redux-thunk — простейший middleware

```
function createThunkMiddleware(extraArgument) {  
  return ({ dispatch, getState }) => next => action => {  
    if (typeof action === 'function') {  
      return action(dispatch, getState, extraArgument);  
    }  
    return next(action);  
  };  
}  
  
const thunk = createThunkMiddleware();  
thunk.withExtraArgument = createThunkMiddleware;  
export default thunk;
```



# Redux-thunk под микроскопом

```
({ dispatch, getState }) => next => action => {
  if (typeof action === 'function') {
    return action(dispatch, getState, extraArgument);
  }
  return next(action);
};
```



# Redux-thunk под микроскопом

```
Redux store  
({ dispatch, getState } ) => next => action => {  
  if (typeof action === 'function') {  
    return action(dispatch, getState, extraArgument);  
  }  
  return next(action);  
};
```



# Redux-thunk под микроскопом

Прокидывает action далее

```
({ dispatch, getState }) => next => action => {
  if (typeof action === 'function') {
    return action(dispatch, getState, extraArgument);
  }
  return next(action);
};
```



# Redux-thunk под микроскопом

```
Текущий action  
({ dispatch, getState }) => next => action => {  
  if (typeof action === 'function') {  
    return action(dispatch, getState, extraArgument);  
  }  
  return next(action);  
};
```



# Redux-thunk под микроскопом

```
({ dispatch, getState } ) => next => action => {  
  if (typeof action === 'function') {  
    return action(dispatch, getState, extraArgument);  
  }  
  return next(action);  
};
```

Обработка кастомных actions



# Обычный action

```
{  
  type: "RECEIVE_VACANCY"  
}
```



# Кастомный action

```
{  
  vacancy: "RECEIVE_VACANCY"  
}
```

✖ ►Uncaught (in promise) Error: Actions may not have an undefined "type" property. Have you misspelled a constant?

at f ([createStore.js:156](#))  
at [registerMiddleware.js:5](#)  
at [index.js:53](#)  
at [batchedMiddleware.js:5](#)

[createStore.js:156](#)



# Middlewares могут обрабатывать кастомные actions

Можно обрабатывать как другие типы данных:  
массивы, функции, так и определенные структуры

Например, redux-thunk — обрабатывает функции



# Middlewares могут обрабатывать кастомные actions

```
const store = createStore(  
  combineReducers({  
    routing: routerReducer,  
    notifications,  
    users,  
    account,  
  }),  
  applyMiddleware(reduxThunk, myOwnMiddleware, logger)  
);
```



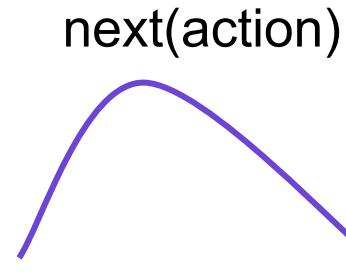
# Middlewares могут обрабатывать кастомные actions

```
const store = createStore(  
  combineReducers({  
    routing: routerReducer,  
    notifications,  
    users,  
    account,  
  }),  
  applyMiddleware(reduxThunk, myOwnMiddleware, logger)  
);
```



# Middlewares могут обрабатывать кастомные actions

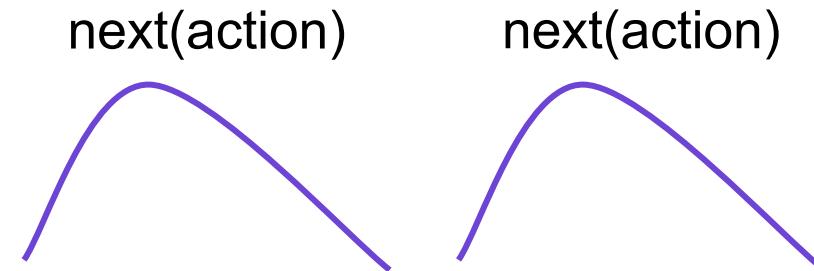
```
const store = createStore(  
  combineReducers({  
    routing: routerReducer,  
    notifications,  
    users,  
    account,  
  }),  
  applyMiddleware(reduxThunk, myOwnMiddleware, logger)  
);
```





# Middlewares могут обрабатывать кастомные actions

```
const store = createStore(  
  combineReducers({  
    routing: routerReducer,  
    notifications,  
    users,  
    account,  
  }),  
  applyMiddleware(reduxThunk, myOwnMiddleware, logger)  
);
```





# Middlewares могут обрабатывать кастомные actions

```
const store = createStore(  
  combineReducers({  
    routing: routerReducer,  
    notifications,  
    users,  
    account,  
  }),  
  applyMiddleware(reduxThunk, myOwnMiddleware, logger)  
);
```

next(action)      next(action)      next(action)

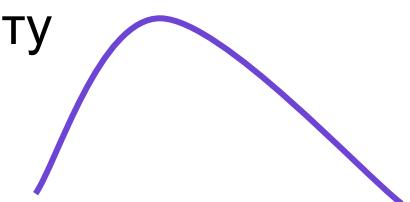
Reducers, redux store



# Middlewares могут обрабатывать кастомные actions

```
const store = createStore(  
  combineReducers({  
    routing: routerReducer,  
    notifications,  
    users,  
    account,  
  }),  
  applyMiddleware(reduxThunk, myOwnMiddleware, logger)  
);
```

Только здесь проверяется  
соответствие action нужному  
формату



Reducers, redux store

```
const app = new Koa();
app.use(async (ctx, next) => {
  var start = new Date;
  await next();
  ctx.set('X-Response-Time', new Date - start + 'ms');
});

app.use(async (ctx, next) => {
  try {
    await next();
  } catch (err) {
    console.error(err);
    ctx.body = { reason: err.message };
    ctx.status = err.status || 500;
  }
});
```

```
const app = new Koa();
app.use(async (ctx, next) => {
  var start = new Date;
  await next();
  ctx.set('X-Response-Time', new Date - start + 'ms');
});

app.use(async (ctx, next) => {
  try {
    await next();
  } catch (err) {
    console.error(err);
    ctx.body = { reason: err.message };
    ctx.status = err.status || 500;
  }
});
```

```
const thunk = ({ dispatch, getState }) => next => action => {
  if (typeof action === 'function') {
    return action(dispatch, getState);
  }
  return next(action);
};
```

```
const logger = ({ dispatch, getState }) => next => action => {
  console.log(getState());
  console.log(action);
  next(action);
  console.log(getState());
};
```

```
const thunk = ({ dispatch, getState }) => next => action => {
  if (typeof action === 'function') {
    return action(dispatch, getState);
  }
  return next(action);
};
```

```
const logger = ({ dispatch, getState }) => next => action => {
  console.log(getState());
  console.log(action);
  next(action);
  console.log(getState());
};
```



# Особенности



Всегда можно использовать композицию из middlewares,  
чтобы выполнить свою задачу



# Особенности



Всегда можно использовать композицию из middlewares, чтобы выполнить свою задачу



Из-за зависимости логики от порядка вставки можно получить неприятные сайд-эффекты. Нужно внимательно следить за очередностью



Окей, а зачем middleware, кроме  
асинхронных запросов?



React-components view

React-redux  
connect

Business logic: action creators via redux-thunk

Actions

Data logic: reducers



# Логирование

Простейший случай:

---

```
const logger = ({ dispatch, getState }) => next => action => {  
  console.log(getState());  
  console.log(action);  
  next(action);  
  console.log(getState());  
};
```



# Логирование

Redux-logger: <https://github.com/evgenyrodionov/redux-logger>

```
prev state                                                 console.js:35
▶ {account: {...}, permissions: {...}, vacanciesSuggest: {...}, location: {...}, records: Array(0), ...}

action      ▶ {type: "ats/tags/RECEIVE", payload: {...}} ⓘ   console.js:35
    ▶ payload:
        ▶ 2: {id: 2, name: "Кадровый резерв"}
        ▶ 27: {id: 27, name: "HR"}
        ▶ 28: {id: 28, name: "Product"}
        ▶ 29: {id: 29, name: "Recruitment"}
        ▶ 30: {id: 30, name: "Java"}
        ▶ 31: {id: 31, name: "Talantix"}
        ▶ 65: {id: 65, name: "моб тестировщик"}
        ▶ 114: {id: 114, name: "тестирование безопасности"}
        ▶ 115: {id: 115, name: "фронтенд"}
        ▶ 120: {id: 120, name: "бренд-менеджер"}
        ▶ 121: {id: 121, name: "Frontend"}
        ▶ 284: {id: 284, name: "сейл/Москва–Ярославль"}
        ▶ __proto__: Object
        type: "ats/tags/RECEIVE"
    ▶ __proto__: Object

next state                                                 console.js:35
▶ {account: {...}, permissions: {...}, vacanciesSuggest: {...}, location: {...}, records: Array(0), ...}
```



# Логирование и обработка исключений

Raven: <https://github.com/getsentry/raven-js> +  
<https://github.com/captbaritone/raven-for-redux>

>-	redux-action	@@router/LOCATION_CHANGE
>-	redux-action	module/CHECKOUT
>-	redux-action	menu/switch
>-	redux-action	restrictedData/CLEAR
⚠	exception	TypeError: e.ref.container is undefined



# Кэширование

```
const storageMiddleware = (selector = state => state, onCatch) => {
  return ({ getState, dispatch }) => next => action => {
    next(action);
    const newState = selector(getState());
    Object.keys(newState)
      .filter(key => cashedState[key] !== newState[key])
      .forEach(saveDataToLocalStorage);
    cashedState = newState;
  };
};
```



# Кэширование

```
const storageMiddleware = (selector = state => state, onCatch) => {
  return ({ getState, dispatch }) => next => action => {
    next(action);  
      
    Action прошел дальше до store
    const newState = selector(getState());
    Object.keys(newState)
      .filter(key => cashedState[key] !== newState[key])
      .forEach(saveDataToLocalStorage);
    cashedState = newState;
  };
};
```



# Кэширование

```
const storageMiddleware = (selector = state => state, onCatch) => {
  return ({ getState, dispatch }) => next => action => {
    next(action); —————> Action прошел дальше до store
    const newState = selector(getState());
    Object.keys(newState)
      .filter(key => cashedState[key] !== newState[key])
      .forEach(saveDataToLocalStorage);
    cashedState = newState;
  };
};
```

Action прошел дальше до store

Сохранили  
изменившиеся  
поля



# Кэширование

```
const storageMiddleware = (selector = state => state, onCatch) => {  
  return ({ getState, dispatch }) => next => action => {  
    next(action); → Action прошел дальше до store  
    const newState = selector(getState());  
    Object.keys(newState)  
      .filter(key => cashedState[key] !== newState[key])  
      .forEach(saveDataToLocalStorage);  
  
    cashedState = newState; → Сменили кэш, для  
    }; } следующего сравнения  
};
```



# Особенность



Middlewares выполняют логику как до передачи action в store, так и после



# Особенность



Middlewares выполняют логику как до передачи action в store, так и после



При использовании middlewares вы не можете гарантировать, что передаваемый action дойдет до store

React-components view

React-redux  
connect

Business logic: action creators via redux-thunk

Actions

Data logic: reducers

# React-components view

React-redux  
connect

Async action creators  
via redux-thunk

Sync action creators

Actions

Middlewares: redux-thunk, redux-logger

Actions

Reducers

## React-components view

React-redux  
connect

Async action creators  
via redux-thunk

Sync action creators

Actions

Middlewares: redux-thunk, redux-logger

Actions

Reducers

Async action creators  
via redux-thunk

Sync action creators

Actions

Middlewares: redux-thunk, redux-logger

Actions

Reducers

Async action creators  
via redux-thunk

Sync action creators

Actions

Logic middleware:  
redux-thunk

Auxiliary middleware:  
1) Redux-logger  
2) Raven middleware

Actions

Reducers

Async action creators  
via redux-thunk

Sync action creators

Actions

Logic middleware:  
redux-thunk

Auxiliary middleware:  
1) Redux-logger  
2) Raven middleware

Actions

Reducers



# Подготовка данных для reducers





Мои вакансии

## Frontend-разработчик

Неразобранные	1
Подумать	1
Тестовое задание	1
Пригласить на интервью	9
Телефонное интервью	● 1/6
Оценка	0
Интервью	3
Групповое собеседование	0
Предложение о работе	0
Выход на работу	1
Отказ	174



Все кандидаты

С меткой



Мостовой Никита

24 года

Frontend-разработчик

HeadHunter

12:27

Подумать

Отказать

Прикрепить к вакансии

Создать событие



## Мостовой Никита Юрьевич

Мужчина, 24 года, Frontend-разработчик, HeadHunter



## Добавить метку

Гражданство: Россия

Регион: Москва

Мобильный телефон: +7 (965) 297-03-79

Email: xnimor@ya.ru

Skype: xnimorz

LinkedIn: http://linkedin.com/

Free-lance: http://free-lance.ru/users/

Мой круг: http://nikita-mostovoy.moikrug.ru/

Резюме НН

Комментарии: 4

Файлы

В вакансиях: 3

## JavaScript Разработчик

11 апреля 2017, посмотреть на HeadHunter

Опыт работы 3 года 5 месяцев



Мои вакансии

Frontend-разработчик

Неразобранные	1
Подумать	1
Тестовое задание	1
Пригласить на интервью	9
Телефонное интервью	● 1/6
Оценка	0
Интервью	3
Групповое собеседование	0
Предложение о работе	0
Выход на работу	1
Отказ	174



Все кандидаты

С меткой



Мостовой Никита

24 года

Frontend-разработчик

HeadHunter

12:27

Подумать

Отказать

Прикрепить к вакансии

Создать событие



## Мостовой Никита Юрьевич

Мужчина, 24 года, Frontend-разработчик, HeadHunter



## Добавить метку

Гражданство: Россия

Регион: Москва

Мобильный телефон: +7 (965) 297-03-79

Email: xnimor@ya.ru

Skype: xnimorz

LinkedIn: http://linkedin.com/

Free-lance: http://free-lance.ru/users/

Мой круг: http://nikita-mostovoy.moikrug.ru/

Резюме НН

Комментарии: 4

Файлы

В вакансиях: 3

## JavaScript Разработчик

11 апреля 2017, посмотреть на HeadHunter

Опыт работы 3 года 5 месяцев



Мои вакансии

## Frontend-разработчик

Неразобранные	1
Подумать	1
Тестовое задание	1
Пригласить на интервью	9
Телефонное интервью	1/6
Оценка	0
Интервью	3
Групповое собеседование	0
Предложение о работе	0
Выход на работу	1
Отказ	174



Все кандидаты

С меткой



Мостовой Никита

24 года

Frontend-разработчик

HeadHunter

12:27

Подумать

Отказать

Прикрепить к вакансии

Создать событие



## Мостовой Никита Юрьевич

Мужчина, 24 года, Frontend-разработчик, HeadHunter



## Добавить метку

Гражданство: Россия

Регион: Москва

Мобильный телефон: +7 (965) 297-03-79

Email: xnimor@ya.ru

Skype: xnimorz

LinkedIn: http://linkedin.com/

Free-lance: http://free-lance.ru/users/

Мой круг: http://nikita-mostovoy.moikrug.ru/

Резюме НН

Комментарии: 4

Файлы

В вакансиях: 3

## JavaScript Разработчик

11 апреля 2017, посмотреть на HeadHunter

Опыт работы 3 года 5 месяцев



Мои вакансии

Frontend-разработчик

Неразобранные

1

Подумать

1

Тестовое задание

1

Пригласить на интервью

9

Телефонное интервью

1/6

Оценка

0

Интервью

3

Групповое собеседование

0

Предложение о работе

0

Выход на работу

1

Отказ

174



Все кандидаты

С меткой

Мостовой Никита

12:27

24 года

Frontend-разработчик

HeadHunter

Подумать

Отказать

Прикрепить к вакансии

Создать событие



## Мостовой Никита Юрьевич

Мужчина, 24 года, Frontend-разработчик, HeadHunter



## Добавить метку

Гражданство: Россия

Регион: Москва

Мобильный телефон: +7 (965) 297-03-79

Email: xnimor@ya.ru

Skype: xnimorz

LinkedIn: http://linkedin.com/

Free-lance: http://free-lance.ru/users/

Мой круг: http://nikita-mostovoy.moikrug.ru/

Резюме НН

Комментарии: 4

Файлы

В вакансиях: 3

## JavaScript Разработчик

11 апреля 2017, посмотреть на HeadHunter

Опыт работы 3 года 5 месяцев



Мои вакансии

Frontend-разработчик

Неразобранные

Подумать

Тестовое задание

Пригласить на интервью

Телефонное интервью

1

1

1

9

1/6

Оценка

0

Интервью

3

Групповое собеседование

0

Предложение о работе

0

Выход на работу

1

Отказ

174



Все кандидаты

С меткой

Мостовой Никита

12:27

24 года

Frontend-разработчик

HeadHunter

Подумать

Отказать

Прикрепить к вакансии

Создать событие



## Мостовой Никита Юрьевич

Мужчина, 24 года, Frontend-разработчик, HeadHunter



Добавить метку

Гражданство: Россия

Регион: Москва

Мобильный телефон: +7 (965) 297-03-79

Email: xnimor@ya.ru

Skype: xnimorz

LinkedIn: http://linkedin.com/

Free-lance: http://free-lance.ru/users/

Мой круг: http://nikita-mostovoy.moikrug.ru/

Резюме НН

Комментарии: 4

Файлы

В вакансиях: 3

## JavaScript Разработчик

11 апреля 2017, посмотреть на HeadHunter

Опыт работы 3 года 5 месяцев



# Данные для отображения страницы

Вакансия

Воркфлоу

Кандидаты

Резюме



# Нормализованные данные в запросе

```
vacancy: {  
    "1": {name: "FrontEnd разработчик", workflow: [1,2]}  
},  
workflow: {  
    "1": {name: "Входящие", unread: 1, candidates: [9, 10, 12]},  
    "2": {name: "Интервью", unread: 2, candidates: [2, 5, 6]}  
},  
candidates: {  
    "9": {name: "Конь Боджек", resumes: [1, 5]},  
    "10": {name: "Здесь может быть ваше имя" }  
    ...  
}
```



# Нормализованные данные в запросе

```
vacancy: {  
    "1": {name: "FrontEnd разработчик", workflow: [1,2]}  
},  
  
workflow: {  
    "1": {name: "Входящие", unread: 1, candidates: [9, 10, 12]},  
    "2": {name: "Интервью", unread: 2, candidates: [2, 5, 6]}  
},  
  
candidates: {  
    "9": {name: "Конь Боджек", resumes: [1, 5]},  
    "10": {name: "Здесь может быть ваше имя" }  
    ...  
}
```



# Один action для всего?

```
function vacancy(state, action) {  
  switch(action.type) {  
    case "RECEIVE_ALL": return action.payload.vacancy;  
    default: return state;  
  }  
}  
  
function candidates(state, action) {  
  switch(action.type) {  
    case "RECEIVE_ALL": return action.payload.candidates;  
    default: return state;  
  }  
}
```



# Один action для всего?

```
function vacancy(state, action) {  
  switch(action.type) {  
    case "RECEIVE_ALL": return action.payload.vacancy;  
    default: return state;  
  }  
}  
  
function candidates(state, action) {  
  switch(action.type) {  
    case "RECEIVE_ALL": return action.payload.candidates;  
    default: return state;  
  }  
}
```



# Один action для всего?

```
function vacancy(state, action) {  
  switch(action.type) {  
    case "RECEIVE_ALL": return action.payload.vacancy;  
    default: return state;  
  }  
}  
  
function candidates(state, action) {  
  switch(action.type) {  
    case "RECEIVE_ALL": return action.payload.candidates;  
    default: return state;  
  }  
}
```



# Один action для всего?

Сложнее дебажить

---

Сложнее отслеживать все связи

---

Сложнее покрывать тестами

---





# Много actions!

```
dispatch({type: "RECEIVE_VACANCY", payload: data.vacancy});  
dispatch({type: "RECEIVE_CANDIDATES", payload:  
data.candidates});  
dispatch({type: "RECEIVE_WORKFLOW", payload: data.workflow});
```



# Много actions!

```
dispatch({type: "RECEIVE_VACANCY", payload: data.vacancy});  
dispatch({type: "RECEIVE_CANDIDATES", payload:  
data.candidates});  
dispatch({type: "RECEIVE_WORKFLOW", payload: data.workflow});
```



# Много actions!

```
dispatch({type: "RECEIVE_VACANCY", payload: data.vacancy});  
dispatch({type: "RECEIVE_CANDIDATES", payload:  
data.candidates});  
dispatch({type: "RECEIVE_WORKFLOW", payload: data.workflow});
```

После каждого action интерфейс будет перерисовываться  
с еще не заполненным до конца store





# Batched actions

```
dispatch([
  {type: "RECEIVE_VACANCY", payload: data.vacancy},
  {type: "RECEIVE_WORKFLOW", payload: data.workflow},
  {type: "RECEIVE_CANDIDATES", payload: data.candidates}
]);
```



# Batched actions

```
dispatch([
  {type: "RECEIVE_VACANCY", payload: data.vacancy},
  {type: "RECEIVE_WORKFLOW", payload: data.workflow},
  {type: "RECEIVE_CANDIDATES", payload: data.candidates}
]);
```



# Batched actions

```
const enableBatching = (rootReducer) => (state, action) => {
  if (action.type === BATCH) {
    return action.payload.reduce(rootReducer, state);
  }
  return rootReducer(state, action);
};

const batchedMiddleware = ({ dispatch }) => next => action => {
  if (!Array.isArray(action)) {
    return next(action);
  }
  dispatch(batch(action));
};
```



# Batched actions

```
const enableBatching = (rootReducer) => (state, action) => {  
  if (action.type === BATCH) {  
    return action.payload.reduce(rootReducer, state);  
  }  
  return rootReducer(state, action);  
};  
  
const batchedMiddleware = ({ dispatch }) => next => action => {  
  if (!Array.isArray(action)) {  
    return next(action);  
  }  
  dispatch(batch(action));  
};
```

Новый root reducer



# Batched actions

```
const enableBatching = (rootReducer) => (state, action) => {  
  if (action.type === BATCH) {  
    return action.payload.reduce(rootReducer, state);  
  }  
  return rootReducer(state, action);  
};  
  
const batchedMiddleware = ({ dispatch }) => next => action => {  
  if (!Array.isArray(action)) {  
    return next(action);  
  }  
  dispatch(batch(action));  
};
```

Новый root reducer

Batched middleware



# Batched actions

```
const batchedMiddleware = ({ dispatch }) => next => action => {  
  if (!Array.isArray(action)) {  
    return next(action);  
  }  
  dispatch(batch(action));  
};
```



# Batched actions

```
const batchedMiddleware = ({ dispatch }) => next => action => {
  if (!Array.isArray(action)) {
    return next(action);
  }
  dispatch(batch(action));
};
```



# Batched actions

```
dispatch([
  {type: "RECEIVE_VACANCY", payload: data.vacancy},
  {type: "RECEIVE_WORKFLOW", payload: data.workflow},
  {type: "RECEIVE_CANDIDATES", payload: data.candidates}
]);
```



# Batched actions

```
dispatch({  
  type: "BATCH",  
  payload: [  
    {type: "RECEIVE_VACANCY", payload: data.vacancy},  
    {type: "RECEIVE_WORKFLOW", payload: data.workflow},  
    {type: "RECEIVE_CANDIDATES", payload: data.candidates}  
  ]  
});
```



# Batched actions

```
const enableBatching = (rootReducer) => (state, action) => {  
  if (action.type === BATCH) {  
    return action.payload.reduce(rootReducer, state);  
  }  
  return rootReducer(state, action);  
};  
  
const batchedMiddleware = ({ dispatch }) => next => action => {  
  if (!Array.isArray(action)) {  
    return next(action);  
  }  
  dispatch(batch(action));  
};
```

Новый root reducer

Batched middleware



# Batched actions

```
dispatch([
  {type: "RECEIVE_VACANCY", payload: data.vacancy},
  {type: "RECEIVE_WORKFLOW", payload: data.workflow},
  {type: "RECEIVE_CANDIDATES", payload: data.candidates}
]);
```



Гибко



# Batched actions

```
dispatch([
  {type: "RECEIVE_VACANCY", payload: data.vacancy},
  {type: "RECEIVE_WORKFLOW", payload: data.workflow},
  {type: "RECEIVE_CANDIDATES", payload: data.candidates}
]);
```



Гибко



Нет лишних перерисовок интерфейса



# Batched actions

```
dispatch([
  {type: "RECEIVE_VACANCY", payload: data.vacancy},
  {type: "RECEIVE_WORKFLOW", payload: data.workflow},
  {type: "RECEIVE_CANDIDATES", payload: data.candidates}
]);
```



Гибко



Нет лишних перерисовок интерфейса



Проще покрывать тестами конкретные actions и reducers



# Batched actions

```
vacancy: {  
    "1": {name: "FrontEnd разработчик", workflow: [1,2]}  
},  
  
workflow: {  
    "1": {name: "Входящие", unread: 1, candidates: [9, 10, 12]},  
    "2": {name: "Интервью", unread: 2, candidates: [2, 5, 6]}  
},  
  
candidates: {  
    "9": {name: "Конь Боджек", resumes: [1, 5]},  
    "10": {name: "Здесь может быть ваше имя" }  
    ...  
}
```

Async action creator via  
redux-thunk

Sync action creator

Actions

Logic middleware:  
redux-thunk

Auxiliary middleware:  
1) Redux-logger  
2) Raven middleware

Batched  
middleware

(Batched) Actions

Reducers

Async action creator via  
redux-thunk

Sync action creator

Actions

Logic middleware:  
redux-thunk

Auxiliary middleware:  
1) Redux-logger  
2) Raven middleware

Batched  
middleware

(Batched) Actions

Reducers



# Универсальный слой для асинхронных запросов и разбора ответов

Дано:

- формат общения фронта и бэка по единому JSON
- любой запрос возвращает унифицированные и нормализованные данные

```
const allInOneMiddleware = ({ getState, dispatch }) => next
=> async action => {
  if (!action.api) {
    next(action);
  }
  dispatch(fetch());
  try {
    const result = await
      axios[action.api.method](action.api.data)
    // Парсим result и делаем dispatch на нужные данные
  } catch(e) {}
};

```

```
const allInOneMiddleware = ({ getState, dispatch }) => next
=> async action => {
  if (!action.api) {
    next(action);
  }
  dispatch(fetch());
  try {
    const result = await
axios[action.api.method](action.api.data)
    // Парсим result и делаем dispatch на нужные данные
  } catch(e) {}
};

```

```
const allInOneMiddleware = ({ getState, dispatch }) => next
=> async action => {
  if (!action.api) {
    next(action);
  }
  dispatch(fetch());
  try {
    const result = await
      axios[action.api.method](action.api.data)
    // Парсим result и делаем dispatch на нужные данные
  } catch(e) {}
};

```



А потом мы попытались это  
сделать...



# Создать событие

Дата и время события

29 мая 2018

с 12:45

—

по 13:55

Кандидат

Выбрано 3 кандидата

Вакансия

Frontend-разработчик (веб-сервисы)

Участники

Выбрано 2 рекрутера

Место проведения

Москва-1

Тема встречи

Собеседование

Комментарий

**B** *I* **Создать событие****Отменить**



Подбор

Мои вакансии

Добавить кандидата

Добавить вакансию

Событие создано X

Кандидаты

Кнопка импорта резюме

Еще ▼

# Календарь

[Создать событие](#)[День](#) [Неделя](#) [Месяц](#)

Май, 2018



Сегодня



пн 21

**вт 22**

ср 23

чт 24

пт 25

сб 26

вс 27

0:00

9:00

10:00

11:00

12:00

13:00

14:00

15:00

9:00  
Мостовой Никита

9:30 2 кандидата



# А потом мы попытались это сделать...

## Переходы по сабмиту

## Нотификации



Мои вакансии

Frontend-разработчик  
(веб-сервисы)

Неразобранные	● 1/3
Подумать	1
Тестовое задание/ уточнения	7
Пригласить на интервью	14
Телефонное интервью	5
Оценка	0
Интервью	3
Групповое собеседование	2
Предложение о работе	0
Выход на работу	1
Отказ	210



Все кандидаты

С меткой

Мостовой Никита  
24 года  
Frontend-разработчик

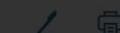
14:10

Тестовое за...

Отказать

Прикрепить к вакансии

Создать событие



## Перевод на этап «Отказ»

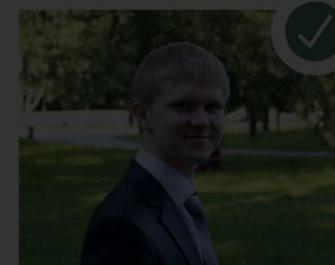
Вы уверены, что хотите отказать кандидату Мостовой Никита? Можно сообщить коллегам о причине отказа в комментарии.

**B** **I**

|

Отменить

Переместить



## JavaScript Разработчик

11 апреля 2017, посмотреть на HeadHunter



Опыт работы: 3 года. Бюджет: 100 000



# А потом мы попытались это сделать...

Переходы по сабмиту

Нотификации

Поп-апы



## Мои вакансии

Frontend-разработчик  
(веб-сервисы)

Неразобранные 3

Подумать 1

Тестовое задание/  
уточнения 7

Пригласить на интервью 9

Телефонное интервью 5

Оценка 0

Интервью 3

Групповое  
собеседование 2

Предложение о работе 0

Выход на работу 1

Отказ 215

Описание Этап

Неразобранные

Подумать

Тестовое задание

Пригласить

Телефонное

Оценка

Интервью

Групповое

Предлож

Выход на работу

Отказ

## Удаление этапа «Подумать»

Выберите этап на который хотите перенести 1 кандидата

- Неразобранные
- Тестовое задание/уточнения
- Пригласить на интервью
- Телефонное интервью
- Оценка
- Интервью
- Групповое собеседование
- Предложение о работе
- Выход на работу
- Отказ

Отменить

Удалить этап



# А потом мы попытались это сделать...

Переходы по сабмиту

Нотификации

Поп-апы

Кастомная логика



# А ПОТОМ МЫ ПОПЫТАЛИСЬ ЭТО СДЕЛАТЬ...

```
if (action.api.type === ACTIONS_TYPES.notify) {  
  // Покажи нотификацию  
}
```



# А ПОТОМ МЫ ПОПЫТАЛИСЬ ЭТО СДЕЛАТЬ...

```
if (action.api.type === ACTIONS_TYPES.notify) {  
    // Покажи нотификацию  
} else if (action.api.type === ACTIONS_TYPES.redirect) {  
    // Покажи другую страницу  
}
```



# А ПОТОМ МЫ ПОПЫТАЛИСЬ ЭТО СДЕЛАТЬ...

```
if (action.api.type === ACTIONS_TYPES.notify) {  
    // Покажи нотификацию  
} else if (action.api.type === ACTIONS_TYPES.redirect) {  
    // Покажи другую страницу  
} else if (action.api.exec) {  
    // Выполнни кастомную логику  
} ...  
...  
...  
...
```







# Но мы извлекли из этого урок

1

Между слоем action creators и middleware работают специальные события — view action или signal action



# Но мы извлекли из этого урок

- 1 Между слоем action creators и middleware работают специальные события — view action или signal action
- 2 Только часть actions, которые не требуют серверной работы, проходят до уровня store (например, валидация формы)

Async action creator via  
redux-thunk

Sync action creator

Actions

Logic middleware:  
redux-thunk

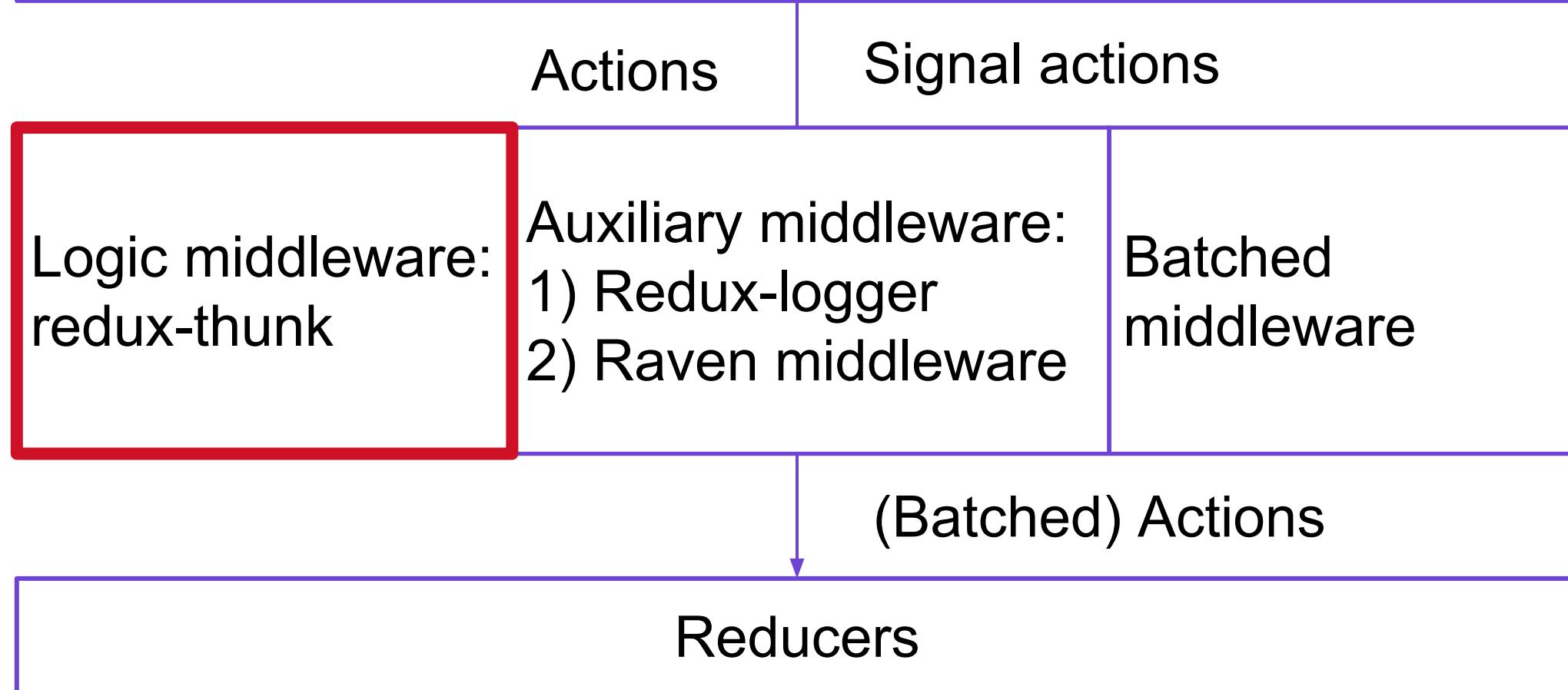
Auxiliary middleware:  
1) Redux-logger  
2) Raven middleware

Batched  
middleware

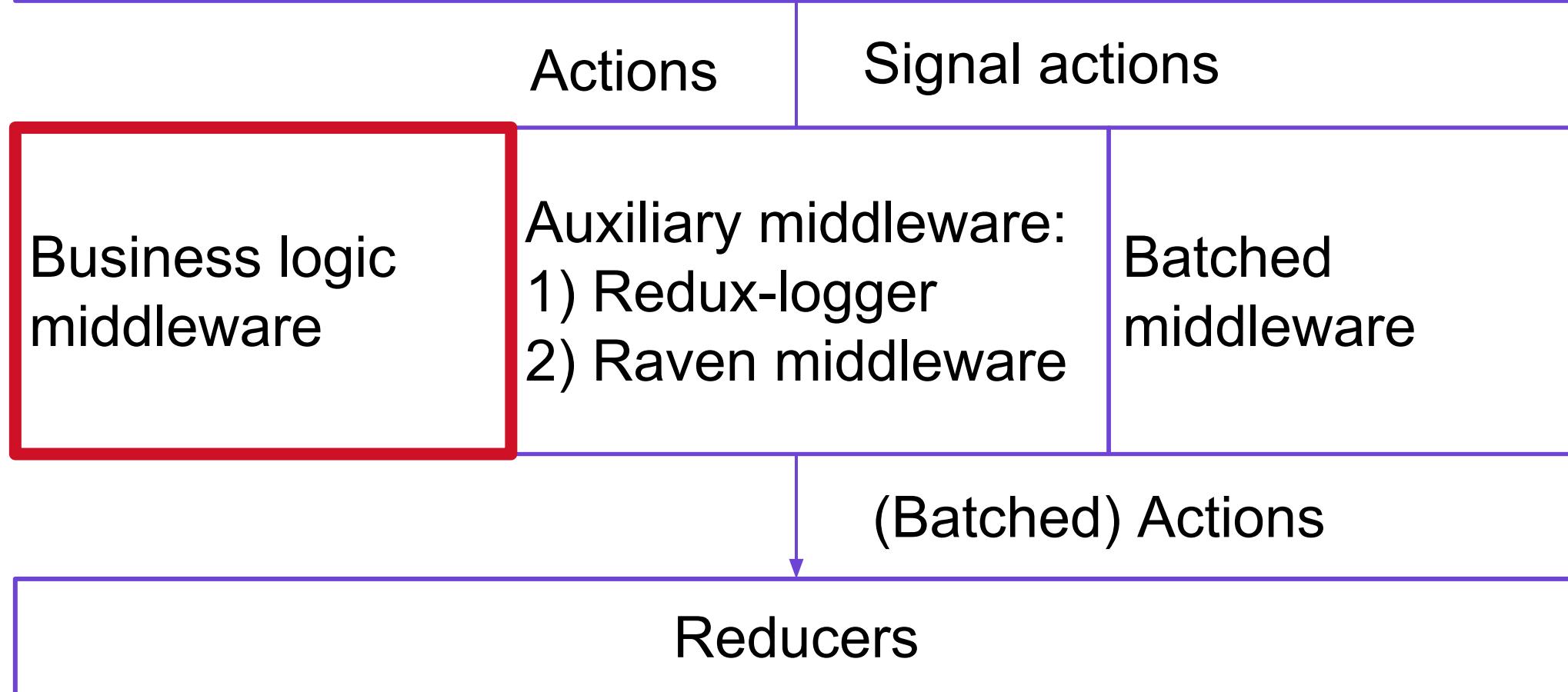
(Batched) Actions

Reducers

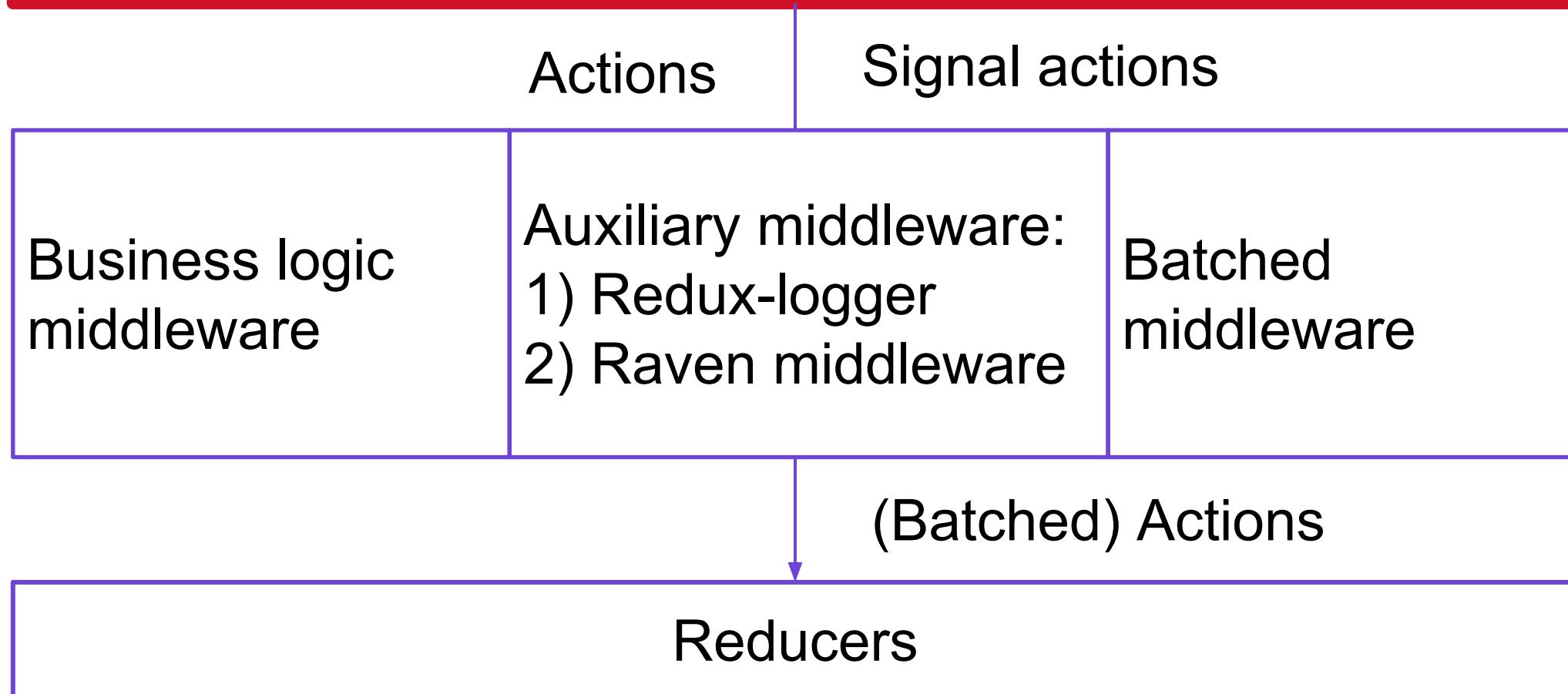
# Sync action creator

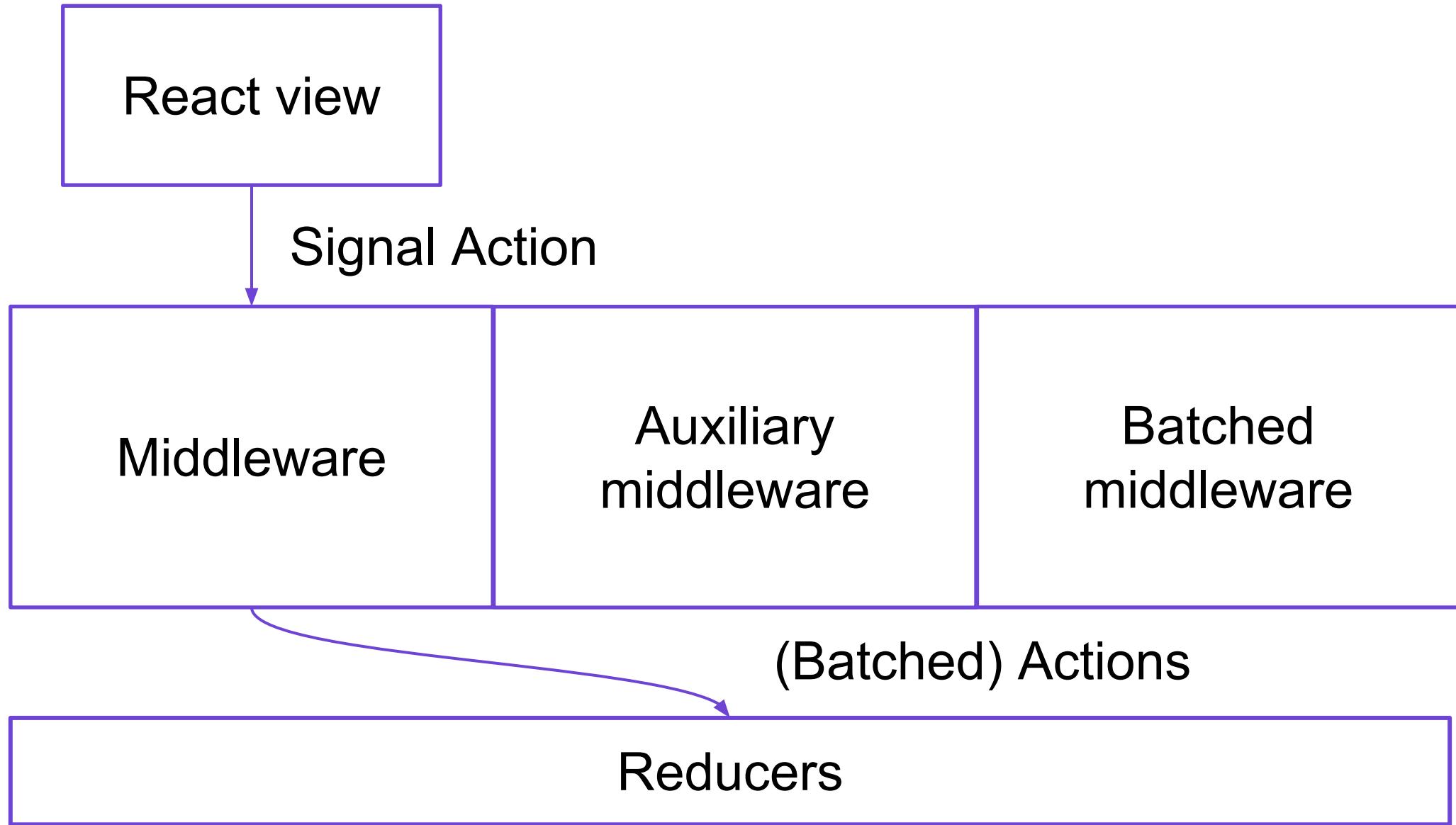


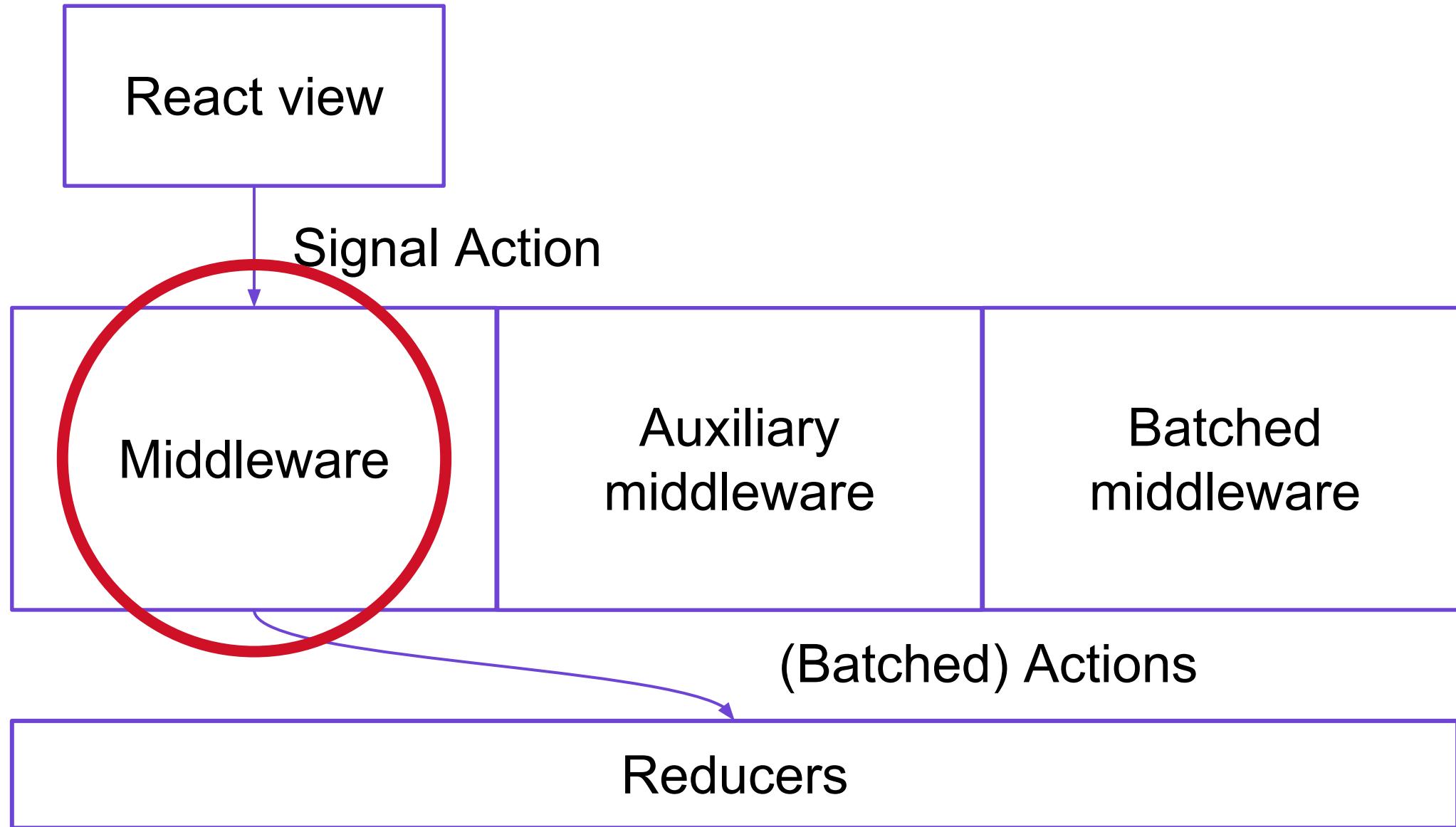
# Sync action creator

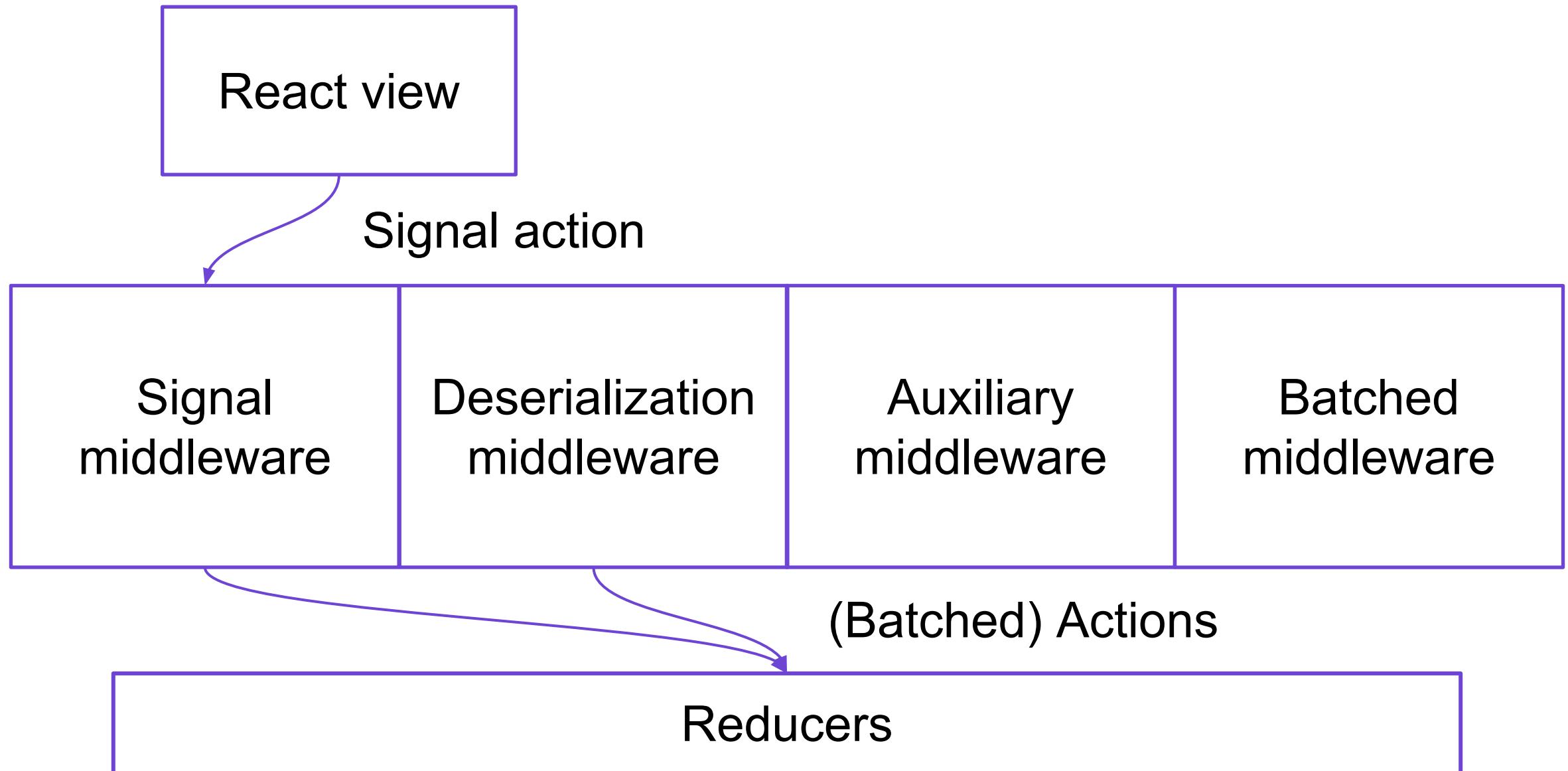


## Sync action creator











# Что получили?

Сильное разделение между view, business и data logic



# Что получили?

Сильное разделение между view, business и data logic

Action creators не содержат весомой логики. Вся бизнес-логика в отдельном слое middleware



# Что получили?

Сильное разделение между view, business и data logic

Action creators не содержат весомой логики. Вся бизнес-логика в отдельном слое middleware

Reducers занимаются только data logic



# Как это выглядит? Обработка сигналов

```
const reactions = {};  
  
const addReactions = signals => reactions = { ...reactions, ...signals };  
  
const signalMiddleware = ({ getState, dispatch }) => next => action => {  
  if (!action.signal) {  
    return next(action);  
  }  
  if (!reactions[action.signal]) {  
    throw new Error(`There is no handler for ${action.signal} signal`);  
  }  
  reactions[action.signal]({ getState, dispatch}, action.payload);  
};
```



# Как это выглядит? Обработка сигналов

```
const reactions = {};  
  
const addReactions = signals => reactions = { ...reactions, ...signals };  
  
const signalMiddleware = ({ getState, dispatch }) => next => action => {  
  if (!action.signal) {  
    return next(action);  
  }  
  if (!reactions[action.signal]) {  
    throw new Error(`There is no handler for ${action.signal} signal`);  
  }  
  reactions[action.signal]({ getState, dispatch}, action.payload);  
};
```



# Как это выглядит? Обработка сигналов

```
const reactions = {};  
  
const addReactions = signals => reactions = { ...reactions, ...signals };  
  
const signalMiddleware = ({ getState, dispatch }) => next => action => {  
  if (!action.signal) {  
    return next(action);  
  }  
  if (!reactions[action.signal]) {  
    throw new Error(`There is no handler for ${action.signal} signal`);  
  }  
  reactions[action.signal]({ getState, dispatch }, action.payload);  
};
```



# Описание бизнес-логики приложения

```
addReactions({  
  [AUTH]: async ({getState, dispatch}, payload) => {  
    const result = await axios.post('/account', payload);  
    dispatch({type: RECEIVE_ACCOUNT, payload: result});  
  },  
  ...  
});
```



# Описание бизнес-логики приложения

```
addReactions({  
  [AUTH]: async ({getState, dispatch}, payload) => {  
    const result = await axios.post('/account', payload);  
    dispatch({type: RECEIVE_ACCOUNT, payload: result});  
  },  
  ...  
});
```



# Описание бизнес-логики приложения

```
addReactions({  
  [AUTH]: async ({getState, dispatch}, payload) => {  
    const result = await axios.post('/account', payload);  
    dispatch({type: RECEIVE_ACCOUNT, payload: result});  
  },  
  ...  
});
```



# Описание бизнес-логики приложения

```
addReactions({  
  [AUTH]: async ({getState, dispatch}, payload) => {  
    const result = await axios.post('/account', payload);  
    dispatch({type: RECEIVE_ACCOUNT, payload: result});  
  },  
  ...  
});
```



# Action to view k business logic layer

```
{  
  signal: AUTH,  
  payload: {...}  
}
```



# Action от business logic layer к data

```
{
```

```
  type: RECEIVE_ACCOUNT,
```

```
  payload: {...}
```

```
}
```



# Deserialization middleware

Формат общения фронта и бэка — единый объект



# Deserialization middleware

Формат общения фронта и бэка — единый объект

Middleware проходится по полям пришедшего объекта и формирует массив actions

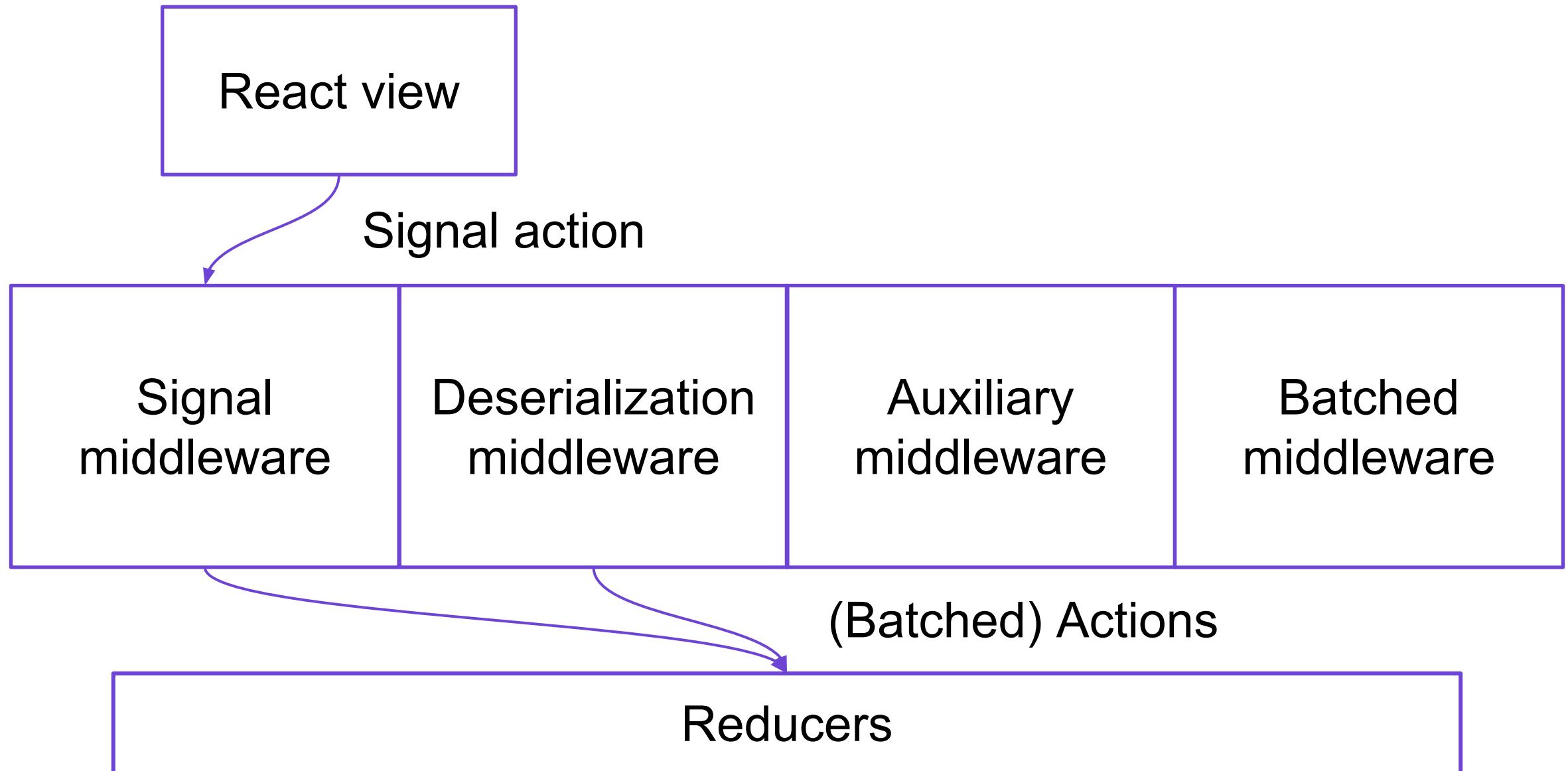


# Deserialization middleware

Формат общения фронта и бэка — единый объект

Middleware проходится по полям пришедшего объекта и формирует массив actions

Middleware делает dispatch





# Подведем итог

1

Middleware — мощный инструмент для разделения кода.  
Имплементируйте бизнес-логику приложения, используя  
middleware



# Подведем итог

1

Middleware — мощный инструмент для разделения кода.  
Имплементируйте бизнес-логику приложения, используя middleware

2

Не создавайте action «сделай все» — это усложняет работу с кодом. Используйте батчи



# Подведем итог

1

Middleware — мощный инструмент для разделения кода. Имплементируйте бизнес-логику приложения, используя middleware

2

Не создавайте action «сделай все» — это усложняет работу с кодом. Используйте батчи

3

Middlewares являются хорошими служебными инструментами для кэширования, логирования и других действий

## Мостовой Никита

<https://goo.gl/PGuIRg> — ссылка  
на все ссылки

Lead Frontend Developer at  
Talantix (HeadHunter)

Twitter: [@xnimorz](https://twitter.com/@xnimorz)

<http://xnim.ru/>

