# Архитектура Flux, взаимодействие с API

| **№ урока:** | 5 | **Курс:** | Архитектура Flux, взаимодействие с API |
|---|---|---|---|

| **Средства обучения:** | Текстовый редактор или IDE, браузер |
|---|---|

## Обзор, цель и назначение урока

В этом уроке мы рассмотрим проблему загрузки и распределения данных среди компонентов, разберемся с архитектурой Flux от Facebook. Также научимся взаимодействовать с внешним API. Начнем писать приложение для работы с задачами.

## Изучив материал данного занятия, учащийся сможет:

- Делать запросы к API
- Настраивать авторизацию
- Использовать Flux

## Содержание урока

1. Данные в приложении
2. MVC, схема, проблемы
3. Однонаправленный поток данных
4. Flux
5. Пример. Покупка товара в интернет-магазине
6. Настройка Google developers console
7. Material-ui
8. Настройка слоя коммуникации с API
9. Пример. Авторизация
10. React-router hooks
11. Пример. Построение Flux архитектуры в приложении: dispatcher, actions, stores, constants

## Резюме

Flux – это архитектура, которую команда Facebook использует при работе с React. Это не фреймворк, или библиотека, это новый архитектурный подход, который дополняет React и принцип однонаправленного потока данных.
Типичная реализация архитектуры Flux может использовать эту библиотеку вместе с классом EventEmitter из NodeJS, чтобы построить событийно-ориентированную систему, которая поможет управлять состоянием приложения.
Facebook предоставляет библиотеку (npm i flux), которая содержит реализацию Dispatcher.
В сущности, Диспетчер – это менеджер всего этого процесса. Это центральный узел вашего приложения. Диспетчер получает на вход действия и рассылает эти действия (и связанные с ними данные) зарегистрированным обработчикам.
Actions – хелперы, упрощающие передачу данных Диспетчеру. Это набор методов, которые вызываются из Представлений (или из любых других мест), чтобы отправить Действия Диспетчеру.
В реализации Facebook Действия различаются по типу — константе, которая посылается вместе с данными действия. В зависимости от типа, Действия могут быть соответствующим образом обработаны в зарегистрированных обработчиках, при этом данные из этих Действий используются как аргументы внутренних методов.
Хранилища в Flux управляют состоянием определенных частей предметной области вашего приложения. На более высоком уровне это означает, что Хранилища хранят данные, методы получения этих данных и зарегистрированные в Диспетчере обработчики Действий.



CyberBionic systematics
CyberBionic Systematics ® 2016
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

## Закрепление материала

1. Что такое архитектура Flux?
2. Как происходит взаимодействие с данными во Flux?
3. Для чего нужны константы в Flux?
4. За что отвечает Dispatcher?
5. Как данные из store попадают в компоненты?

## Самостоятельная деятельность учащегося

**Задание 1: Удаление задач**
*Уровень сложности: низкий*
Нужно реализовать удаление задач из списка (по клику в меню) и при редактировании. Для этого нужно создать методы для API, константы, экшены и обработчики в сторах.

Документация: https://developers.google.com/google-apps/tasks/v1/reference/tasks/delete

**Задание 2: Выделение выбранного списка задач**
*Уровень сложности: низкий*
В меню слева должен подсвечиваться выбранный список. Почитайте в документации react-router о том, как можно это сделать.

**Задание 3: Отображение названия выбранного списка**
*Уровень сложности: средний*
Вверху списка задач должно отображаться название открытого списка.

Документация: https://developers.google.com/google-apps/tasks/v1/reference/tasklists/get

**Задание 4: Отображение названия выбранного списка**
*Уровень сложности: высокий*
При создании (и, соответственно, редактировании) задач должна быть возможность добавить к ним описание и срок выполнения. Используйте компоненты material-ui. Также, подумайте, как это лучше отобразить в приложении при просмотре списка.

Создание задачи: https://developers.google.com/google-apps/tasks/v1/reference/tasks/insert

Изменение задачи: https://developers.google.com/google-apps/tasks/v1/reference/tasks/update

Компонент для выбора даты: http://www.material-ui.com/v0.14.4/#/components/date-picker

**Задание 5: Редактирование и удаление списков**
*Уровень сложности: высокий*
При открытии списка вверху (рядом с кнопкой для добавления задачи) должны быть кнопки для редактирования и удаления открытого списка. Предусмотрите обновления информации в списке в приложение, после ее изменения. Также добавьте диалог с предупреждением при удалении списка.

Изменение списка: https://developers.google.com/google-apps/tasks/v1/reference/tasklists/update

Удаление списка: https://developers.google.com/google-apps/tasks/v1/reference/tasklists/delete

## Рекомендуемые ресурсы

**Все примеры из урока и пояснения к домашним заданиям можно найти тут:**
https://github.com/krambertech/react-essential-course

**Flux**

- Официальный сайт - (https://facebook.github.io/flux/)



CyberBionic systematics
CyberBionic Systematics ® 2016
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

**Google API**

- https://console.developers.google.com/iam-admin/projects
- https://developers.google.com/google-apps/tasks/v1/reference

**Дополнительно**

- Библиотека material-ui (http://material-ui.com)

**Что почитать:**

Официальная документация React - http://facebook.github.io/react/docs

Советую ознакомиться с JavaScript Style Guide - https://github.com/airbnb/javascript



CyberBionic systematics

CyberBionic Systematics ® 2016
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com