

Федеральное государственное бюджетное образовательное учреждение высшего
образования «Московский авиационный институт
(национальный исследовательский университет)»

Институт №3

Системы управления, информатика и электроэнергетика
Кафедра 304: «Вычислительные машины, системы и сети»

Отчёт по курсовой работе
по учебной дисциплине:
Интернет-Технологии

Выполнил:
Кузнецов О.В.
гр. М3О-307Б-17

Проверил:
Титов Ю.П

Москва 2020

Задание на курсовую работу по курсу «Интернет технологии»

Разработать клиент-серверное AJAX приложение.

Серверное BackEnd приложение необходимо разработать на **Node.JS** (необходимо поставить менеджер пакетов npm) с применением фреймворка **Express**.

Для хранения информации на сервере используется документоориентированная NOSQL база данных **MongoDB**.

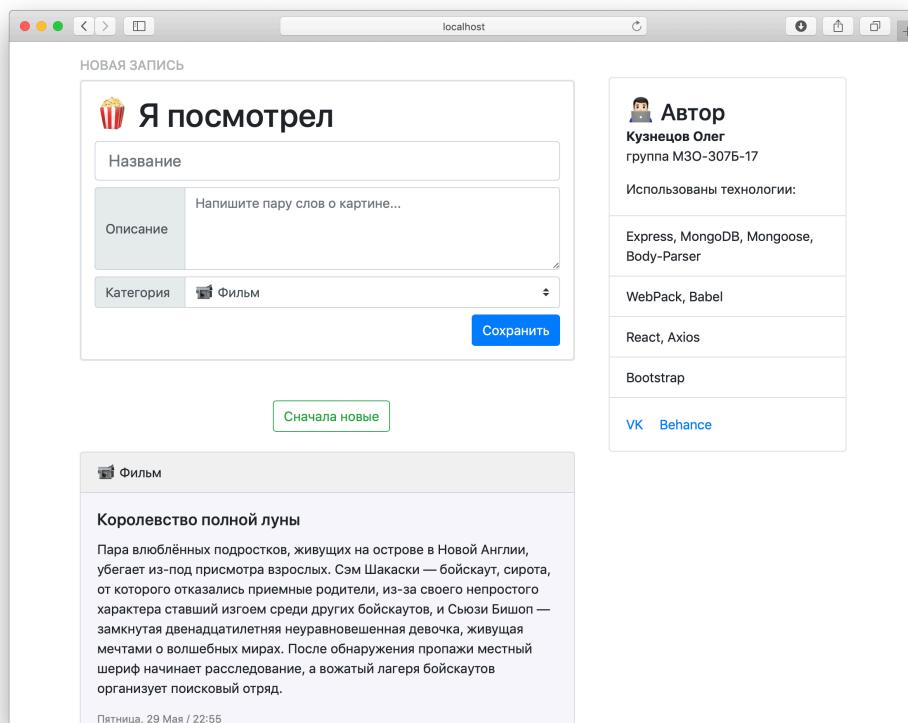
Клиентское приложение пишется с помощью фреймворка **React.JS** с применением CSS для более приятного визуального отображения информации.

В результате клиент-серверное программное приложение должно обеспечивать:

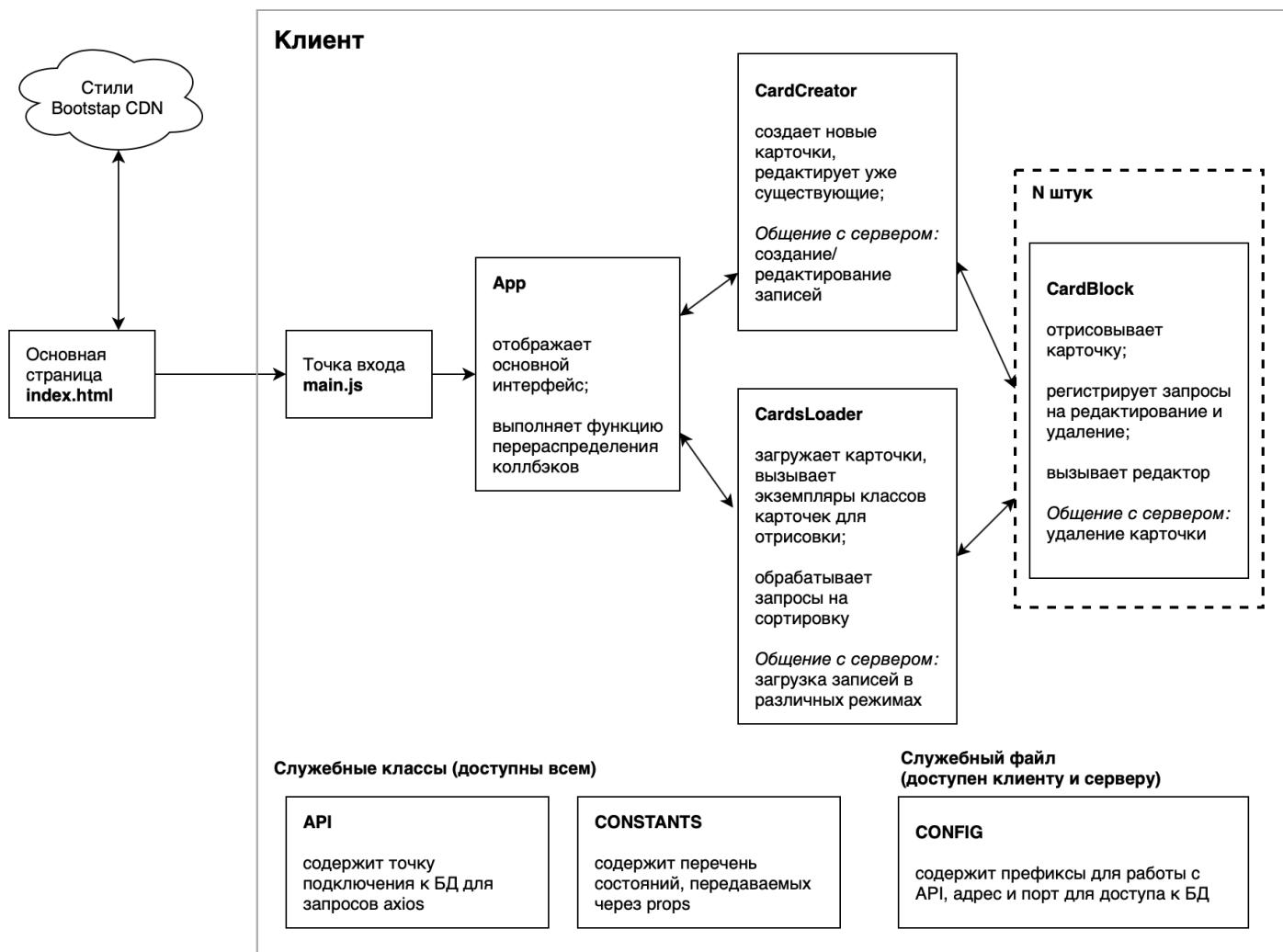
- Возможность подключения множества клиентов к серверу (в идеале к удаленному серверу)
- Загрузки информации о разработчиках и назначении программы
- Возможность добавления новой информации на сервер
- Возможность удаления с сервера необходимой информации
- Возможность вывода на экран клиента информации, хранящейся на сервере с возможностью выбора условий вывода (например, вывести все пары на сегодня) / различных вариантов сортировки данных.

Выбранный вариант:

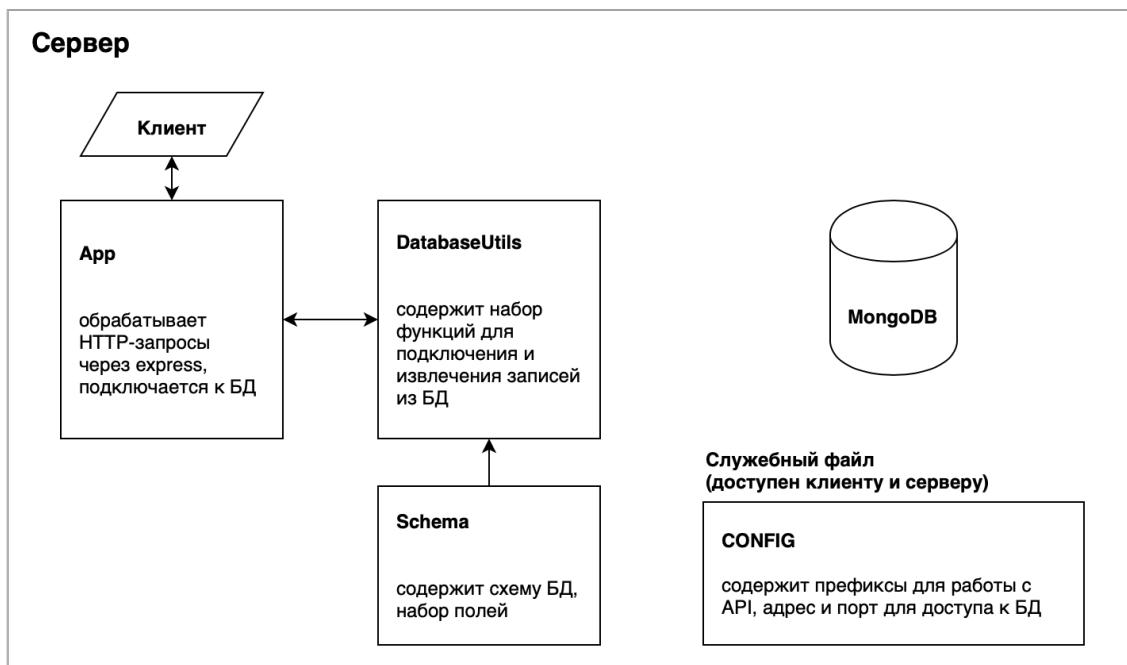
Movies App — приложение для учета просмотренных фильмов / сериалов / роликов.



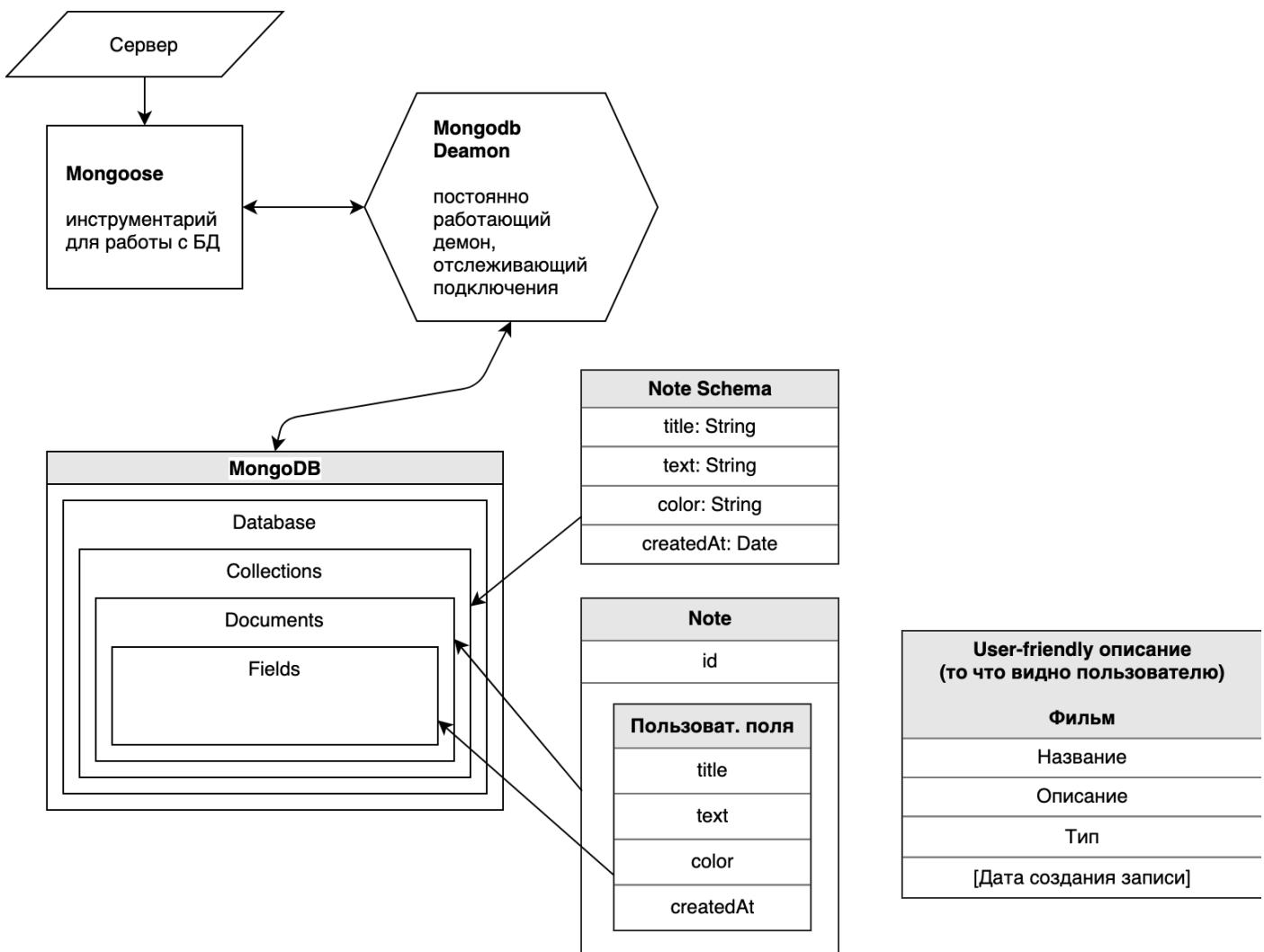
Клиентская часть приложения:



Серверная часть приложения:



База данных:



Пример ответа от сервера при получении записей из БД:

```

CardsLoader.jsx?e373:125
{
  data: Array(2),
  status: 200,
  statusText: "OK",
  headers: {...},
  config: {...}
}

  baseURL: "http://localhost:8080"
  data: undefined
  headers: {Accept: "application/json, text/plain, */*"}
  method: "get"
  timeout: 0
  transformRequest: [f]
  transformResponse: [f]
  url: "http://localhost:8080/notes"
  withCredentials: undefined
  xsrfCookieName: "XSRF-TOKEN"
  xsrfHeaderName: "X-XSRF-TOKEN"
  __proto__: Object
}

data: Array(2)
  0: {_id: "5ed0304cee02bf5e1918eb72", title: "Stranger Things..."}
  1: {_id: "5ed168a7fc045e57d56330fe", title: "Королевство пол...
  length: 2
  __proto__: Array(0)
}

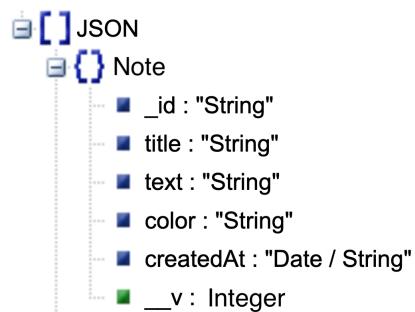
headers:
  content-type: "application/json; charset=utf-8"
  __proto__: Object
status: 200
statusText: "OK"

```

— в случае обращения клиента через сервер к БД, сервер возвращает:

- ответ, содержащий данные в формате JSON (не подвергаются дополнительному форматированию при роутинге на сервере, просто отправляется ответ от БД)
- служебную информацию о подключении, которая в частности содержит код ответа, тип возвращаемых данных, информацию о кодировке и т. д.

Структура JSON-данных:



Пример JSON-данных:

```
1  {
2  {
3      "_id": "5ed0304cee02bf5e1918eb72",
4      "title": "Stranger Things",
5      "text": "Американский научно-фантастический сериал, созданный братьями Дафферами для стримингового сервиса Netflix\n",
6      "color": "📺 Сериал",
7      "createdAt": "2020-06-01T08:29:26.027Z",
8      "__v": 0
9  },
10 {
11     "_id": "5ed168a7fc045e57d56330fe",
12     "title": "Королевство полной луны",
13     "text": "Пара влюблённых подростков, живущих на острове в Новой Англии, убегает из-под присмотра взрослых. Сэм Шакаски – бойскаут, сирота, от которого отказались приемные родители, из-за своего непростого характера ставший изгоем среди других бойскаутов, и Сьюзи Бишоп – замкнутая двенадцатилетняя неуравновешенная девочка, живущая мечтами о волшебных мирах. После обнаружения пропажи местный шериф начинает расследование, а вожатый лагеря бойскаутов организует поисковый отряд.",
14     "color": "📺 Фильм",
15     "createdAt": "2020-05-30T06:27:20.550Z",
16     "__v": 0
17 }
```

Вид при запуске веб-приложения:

The screenshot shows a web browser window with the address bar set to 'localhost'. The main content area displays a heading 'Сдаем курсачи, смотрим сериалы' (Renting course projects, watching series) and a subtitle 'Небольшое SPA-приложение на React для учета любимых фильмов' (A small SPA application on React for keeping track of favorite movies) with a rocket emoji. Below this, there's a note 'Куча кода и немного магии' (A lot of code and a little magic). On the left, a 'НОВАЯ ЗАПИСЬ' (New entry) button is visible, which is highlighted in the middle section. This section contains a form titled 'Я посмотрел' (I watched) with fields for 'Название' (Name), 'Описание' (Description), and 'Категория' (Category) set to 'Фильм' (Movie). A 'Сохранить' (Save) button is at the bottom right. To the right of the form is a sidebar with a profile picture of a person, the name 'Автор' (Author), 'Кузнецов Олег' (Kuznetsov Oleg), and 'группа МЗО-307Б-17' (group MZO-307B-17). Below this, a list of technologies used is shown: Express, MongoDB, Mongoose, Body-Parser, WebPack, Babel, React, Axios, and Bootstrap. At the bottom of the sidebar is a link 'Узнать больше' (Learn more).

Главный экран содержит:

- шапку сайта
- блок добавления записи
- информацию об авторе

Блок добавления записи наряду с добавлением записи без перезагрузки страницы дополнительно демонстрирует функционал динамического изменения контента:

- заголовок изменяется по мере заполнения поля названия кинокартины

The image shows three sequential screenshots of the 'НОВАЯ ЗАПИСЬ' (New entry) form. In the first screenshot, the title field is empty and contains the placeholder 'Название' (Name). In the second screenshot, the user has typed 'Ин' into the title field, and the placeholder is still visible. In the third screenshot, the user has completed the title with 'Интер', and the placeholder is no longer visible, indicating a real-time update.

Лента добавленных записей:

The screenshot shows a web browser window with the address bar set to 'localhost'. The main content area displays a list of three entries, each represented by a card:

- Книга**:
закнутая двенадцатилетняя неуравновешенная девочка, живущая мечтами о волшебных мирах. После обнаружения пропажи местный шериф начинает расследование, а вожатый лагеря бойскаутов организует поисковый отряд.
Пятница, 29 Мая / 22:55
[Удалить](#) [Редактировать](#)
- Фильм**:
Отель «Гранд Будапешт»
Кинофильм Уэса Андерсона по мотивам рассказов Стефана Цвейга с Рэйфом Файнсом и Тони Револори в главных ролях. Действие фильма происходит в вымышленной восточноевропейской стране Зубровке.
Пятница, 29 Мая / 14:32
[Удалить](#) [Редактировать](#)
- Сериал**:
Stranger Things
Американский научно-фантастический сериал, созданный братьями Дафферами для стримингового сервиса Netflix
Пятница, 29 Мая / 13:24
[Удалить](#) [Редактировать](#)

Лента добавленных записей состоит из карточек, каждая из которых включает в себя:

- название
- описание
- категорию
- дату создания / изменения

Для каждой категории предусмотрена отдельная иконка для лучшего визуального восприятия, а дата создания записи, генерируемая в БД в момент ее создания, правильно форматируется.

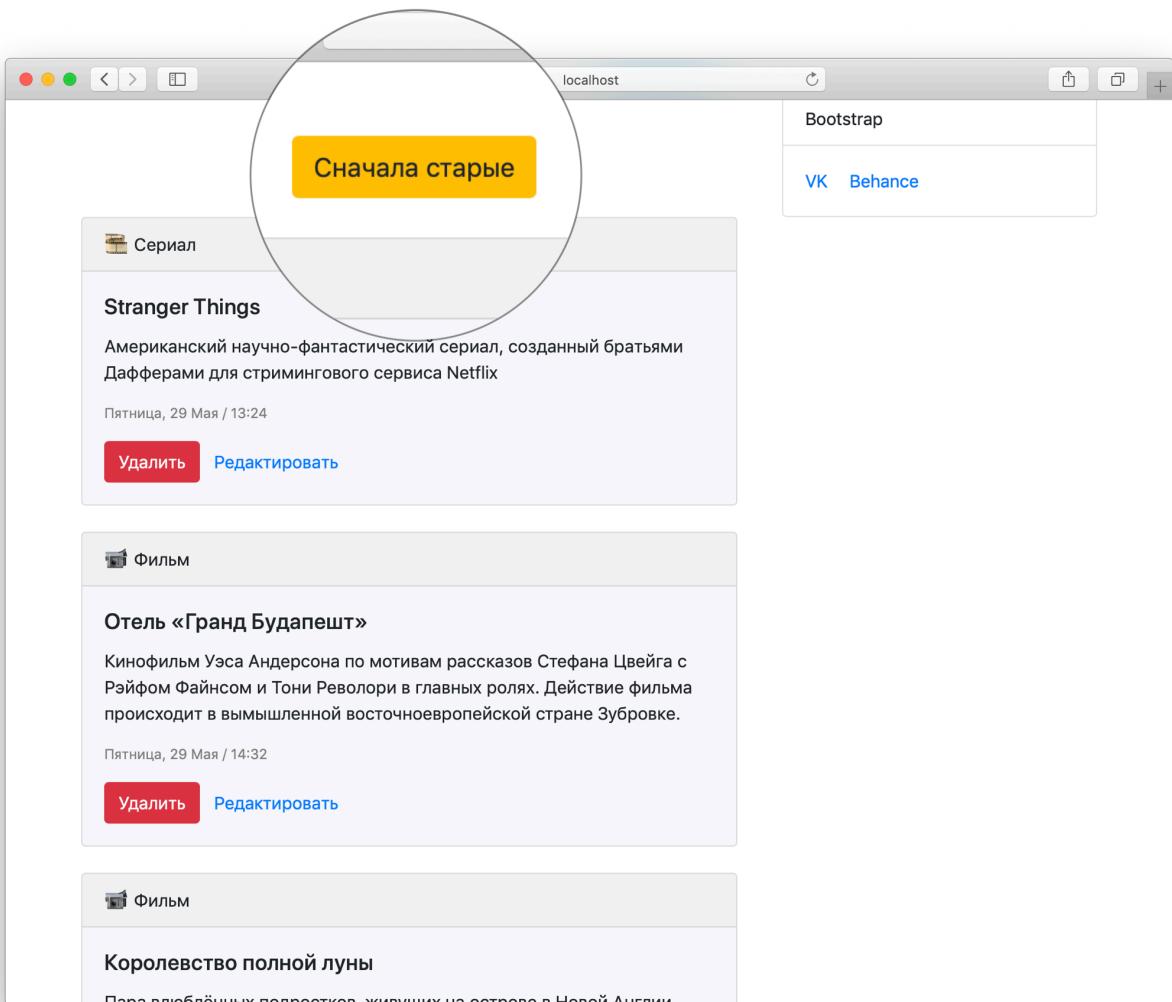
Каждую запись можно отредактировать (*редактирование записи — далее*)

Редактирование записи:

The screenshot shows a web application interface. At the top, there's a browser header with icons for window control and a search bar. The address bar says "localhost". Below the header is a yellow card representing a movie entry. The card has a small icon of a film reel and the word "Фильм". The title "Отель «Гранд Будапешт»" is displayed. A short description follows: "Кинофильм Уэса Андерсона по мотивам рассказов Стефана Цвейга с Рэйфом Файнсом и Тони Револори в главных ролях. Действие фильма происходит в вымышленной восточноевропейской стране Зубровке." Below the description is the date "Пятница, 29 Мая / 14:32". At the bottom of the card are two buttons: a red "Удалить" (Delete) button and a blue "Редактировать" (Edit) button. Below the card, the word "РЕДАКТИРОВАНИЕ" is written in capital letters. Underneath it is a modal dialog box titled "🛠️ Отель «Гранд Будапешт»". Inside the dialog, the title "Отель «Гранд Будапешт»" is shown in a box. To the left of the dialog, there are two columns: "Описание" (Description) containing the movie's plot, and "Категория" (Category) with a dropdown menu set to "Фильм". Below these fields is a checkbox labeled "Не изменять дату" (Do not change the date) with the note "Если изменить дату, записи будет присвоена текущая дата" (If the date is changed, the record will be assigned the current date). At the bottom right of the dialog is a blue "Сохранить" (Save) button.

- При редактировании страницы происходит перерисовка блока редактируемой карточки, к которой добавляется редактор.
- Редактор представляет собой модифицированный класс который уже использовался ранее для создания карточки. Передавая нужные props при вызове класса редактора указывается режим его работы.

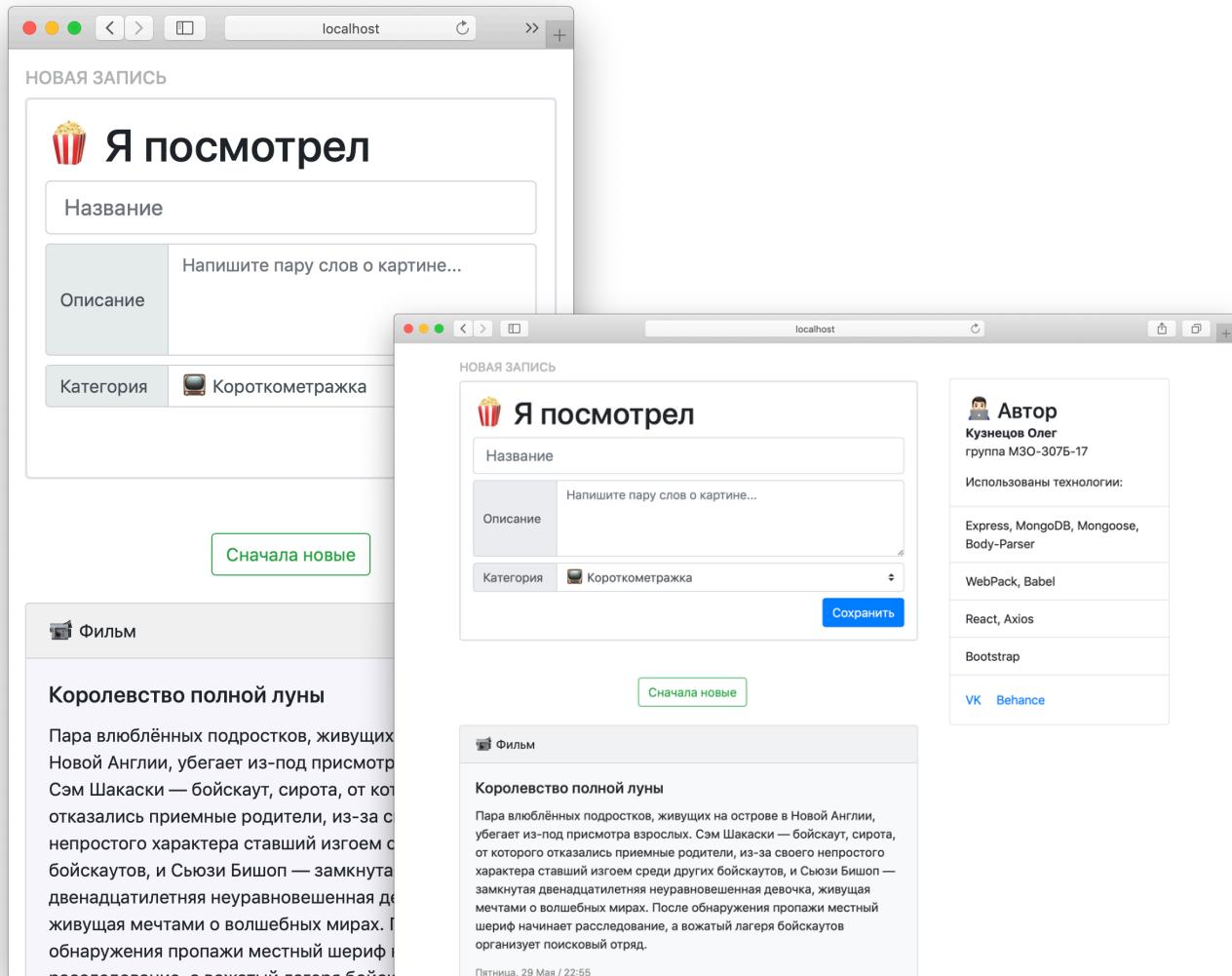
Работа с данными: сортировка:



Реализован функционал работы с данными в виде сортировки:

- при нажатии на кнопку блок отрисовки карточек знает о новом режиме вывода данных и изменяет URL запроса к API БД. Та в свою очередь использует иной параметр метода .sort при обращении к MongoDB.

Адаптивная верстка:



Верстка приложения реализована на Bootstrap. Благодаря применению общего контейнера и уже доступных стилей, в зависимости от размера экрана устройства элементы меняют размер:

```
// Малые девайсы («ландашафтные телефоны», >= 576px)
@media (min-width: 576px) { ... }

// Средние девайсы («таблетки», >= 768px)
@media (min-width: 768px) { ... }

// Большие девайсы (десктопы, >= 992px)
@media (min-width: 992px) { ... }

// Экстрабольшие девайсы (большие десктопы, >= 1200px)
@media (min-width: 1200px) { ... }
```

Клиент

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>🍿 Movies App</title>
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <!-- Bootstrap CSS -->
  <link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
    integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwA0H8WgZl5MYxFFc+NcPb1dKGj7Sk"
    crossorigin="anonymous">
</head>
<body>
  <div id='mount-point'></div>
  <script type="text/javascript" src="build/bundle.js"></script>
</body>
</html>
```

Main.js

```
import React from 'react';
import ReactDOM from 'react-dom';

import App from './components/App.jsx';

ReactDOM.render(
  <App/>,
  document.getElementById('mount-point')
);
```

App.jsx

```
import React from "react";

import CardsLoader from "./CardsLoader.jsx";
import CardCreator from "./CardCreator.jsx";
import constants from "./constants.js";

class App extends React.Component {

  constructor(props) {
    super(props);
    this.state = {
      message: constants.DEFAULT,
      sort: false
    }
    this.handleChange = this.handleChange.bind(this);
  }

  handleChange(value) {
    if (this.state.message != constants.LOAD_CARDS_REQUEST || value == constants.SUCCESS_LOAD) {
      this.setState({message: value});
      console.log(`App:// new message registered: ${value}`);
    }
    if (value == constants.SORT_REQUEST) {
      this.setState({message: value, sort: !this.state.sort});
    }
  }

  render() {
    return (
      <div className="container" style={{maxWidth: "960px"}}>
```

```

        <div className="jumbotron jumbotron-fluid">
            <div className="container">
                <h1 className="display-4 ml-5">Сдаем курсачи, <br/>смотрим сериалы</h1>
                <p className="lead ml-5">Небольшое SPA-приложение на React для учета
        любимых фильмов <img alt="play icon" style="vertical-align: middle;"/></p>
                <p className="ml-5" style={{color: "grey"}}>Куча кода и немного магии</p>
            </div>
        <div className="row">
            <div className="col-sm">
                <h6 style={{color: "#b3b4b5"}}>НОВАЯ ЗАПИСЬ</h6>
                <div className="border rounded" style={{maxWidth: "600px", minWidth: "100px"}><div className="p-3 border rounded"><CardCreator message={this.state.message} parentCallback = {this.handleChange}/> </div> </div>
                    <div className="mt-5"><CardsLoader message={this.state.message} sort={this.state.sort} parentCallback = {this.handleChange}/> </div>
                </div>
            <div class="col-sm">
                <div className="card mt-4 mr-3 ml-3" style={{width: "18rem"}}>
                    <div className="card-body">
                        <h2 className="d-inline"><img alt="camera icon" style="vertical-align: middle;"/></h2><h3 className="card-title d-inline"> Автор</h3>
                        <p className="card-text"><b>Кузнецов Олег</b><br/>группа М30-307Б-17</p>
                        <p className="card-text">Использованы технологии:</p>
                    </div>
                    <ul className="list-group list-group-flush">
                        <li className="list-group-item">Express, MongoDB, Mongoose, Body-Parser</li>
                        <li className="list-group-item">WebPack, Babel</li>
                        <li className="list-group-item">React, Axios</li>
                        <li className="list-group-item">Bootstrap</li>
                    </ul>
                    <div className="card-body">
                        <a href="https://vk.com/okuznetcov" className="card-link">VK</a>
                        <a href="https://www.behance.net/okuznetcov" className="card-link">Behance</a>
                    </div>
                </div>
            </div>
        </div>
    );
}
}

export default App;

```

CardCreator.jsx

```

import React from 'react';
import API from "../api/index";
import regeneratorRuntime from "regenerator-runtime";
import { apiPrefix } from "../../etc/config.json";
import CardsLoader from "./CardsLoader.jsx";
import constants from "./constants.js";

class CardCreator extends React.Component {

    constructor(props) {
        super(props);

        this.state = {
            title: this.props.title,
            text: this.props.text,
            color: this.props.color
        };
        this.inEditMode = false;
        this.id = this.props.id;
        this.changeHandler = this.changeHandler.bind(this);
        this.submitHandler = this.submitHandler.bind(this);
    }
}

```

```

this.changeDate = false;

if (this.id != null)
  this.inEditMode = true;
else
  this.state = {color: ' FILM'};

}

submitHandler(event) {
  event.preventDefault();
  this.saveNote();
  this.setState({ title: '', text: '', color: ''});
}

changeHandler(event) {
  let nam = event.target.name;
  let val = event.target.value;
  console.log(nam);
  console.log(val);
  this.setState({
    [nam]: val
  });

  if (nam == "changeDate")
    this.changeDate = !this.changeDate;
}

render() {

  let now = new Date();

  const editorCheckbox = <div className="text-right form-check mt-2"><input name="changeDate" type="checkbox" onChange={this.changeHandler} className="form-check-input"></input><label className="form-check-label">Не изменять дату</label><br/><small id="passwordHelpInline" class="text-muted">Если изменить дату, записи будет присвоена текущая дата</small></div>;
  const checkboxHelper = <small id="passwordHelpInline" class="text-muted">Must be 8-20 characters long.</small>

  return (
    <div>
      <form onSubmit={this.submitHandler}>
        {this.inEditMode ? <h1>*<span> {this.state.title}</span> : <h1>🍿 Я посмотрел
{this.state.title}</h1>}
        <input
          className="form-control form-control-lg"
          type='text'
          name='title'
          onChange={this.changeHandler}
          placeholder='Название'
          value={this.state.title}
        />

        <div className="input-group mt-2" style={{'minHeight': "100px"}}>
          <div className="input-group-prepend">
            <span className="input-group-text" style={{"minWidth": "110px" }}>Описание</span>
          </div>
          <textarea className="form-control" aria-label="With textarea"
            type='text'
            name='text'
            onChange={this.changeHandler}
            placeholder='Напишите пару слов о картине...'
            value={this.state.text}
          />
        </div>
        <div className="input-group mt-2">
          <div className="input-group-prepend">
            <p className="input-group-text" style={{"minWidth": "110px" }}>Категория</p>
          </div>
          <select name='color' className="custom-select" value={this.state.color} onChange={this.changeHandler}>
            <option value=' FILM'> FILM</option>

```

```

        <option value='📺 Сериял'>📺 Сериял</option>
        <option value='🎬 Короткометражка'>🎬 Короткометражка</option>
        <option value='💻 YouTube / Vimeo'>💻 YouTube / Vimeo</option>
    </select>
</div>
{this.inEditMode ? editorCheckbox : null}
<div className="text-right">
    <input type='submit' className="btn btn-primary mt-2" value="Сохранить"/>
</div>
</form>
</div>
);
}

saveNote() {

let url = "/notes";
console.log(url);

if (this.inEditMode)
    url = `/notes/edit/${this.props.id}`;

if (this.inEditMode && this.changeDate)
    url = `/notes/edit/withoutDateModify/${this.id}`;

var code = 0;

API.post(url, {
    title: this.state.title,
    text: this.state.text,
    color: this.state.color
})
.then(response => {
    this.callback(constants.LOAD_CARDS_REQUEST)
})
.catch(error => {
    this.callback(constants.SEND_CARD_ERR)
});
}

callback(value) {
    this.props.parentCallback(value);
}
}

export default CardCreator;

```

CardsLoader.jsx

```

import React from "react";

import API from "../api/index";
import CardBlock from "./CardBlock.jsx";
import regeneratorRuntime from "regenerator-runtime";
import { apiPrefix } from '../../../../../etc/config.json';
import constants from "./constants.js";

class CardsLoader extends React.Component {

constructor(props) {
    super(props);

    this.state = {
        isLoading: true,
        data: [],
        sorting: this.props.sort
    };
}

console.log(`CardsLoader: firstload / sort: ${this.state.sorting}`);

this.handleCard = this.handleCard.bind(this);

```

```

    this.handleSort= this.handleSort.bind(this);
}

componentWillReceiveProps(nextProps) {
  if (nextProps.message == constants.LOAD_CARDS_REQUEST) {
    console.log("updateRequest");
    this.getNotes();
  } else if (nextProps.message == constants.SORT_REQUEST) {
    this.setState({sorting: nextProps.sort});
    this.callback(constants.LOAD_CARDS_REQUEST);
  }
}

handleCard(value) {
  let action = value.split('/')[0];
  let id = value.split('/')[1];

  if (action == "DELETE") {
    console.log(id);
    this.deleteCard(id);
  } else if (action == "EDIT") {
    this.callback(constants.LOAD_CARDS_REQUEST);
  }
}

handleSort() {
  console.log("SortButton Pressed. Sending request");
  this.callback(constants.SORT_REQUEST);
}

deleteCard(id) {
  API.delete(`/notes/${id}`).then(res => {
    console.log(res);
  })
}

render() {
  const {isLoading, data} = this.state;
  let sortClass = "btn btn-outline-";

  const loadingMessage = (
    <center>
      <div>
        <div className="spinner-border mt-5" role="status">
          <span className="sr-only mt-5">Loading...</span>
        </div>
        <p style={{color: "grey"}}>Admins do not drink chocolate</p>
      </div>
    </center>;
  )

  if (this.state.sorting)
    sortClass = sortClass + "warning";
  else
    sortClass = sortClass + "success";

  let cards = data.map(s => ( <CardBlock data={s} parentCallback={this.handleCard}> ) );

  return (
    <div>
      <div className="text-center">
        <button className={sortClass} onClick={this.handleSort}>
          {this.state.sorting ? 'Сначала старые' : 'Сначала новые'}
        </button>
      </div>
      {isLoading ? loadingMessage : cards}
    </div>
  );
}

```

```

async componentDidMount() {
    this.getNotes()
}

async getNotes() {
    let url = `/notes`;

    if (this.state.sorting) {
        url = url + `/sort/reversed`;
        console.log(url);
    }

    let userData = await API.get(url);

    const data = userData.data;
    this.prevQty = data.length;

    this.setState({
        ...this.state, ...{
            isLoading: false,
            data
        }
    });

    this.callback(constants.SUCCESS_LOAD);

    console.log(`GET COMPLETE / Sorting: ${this.state.sorting} / URL: ${url}`);
    console.log(data);
}

callback(value) {
    this.props.parentCallback(value);
}
}

export default CardsLoader;

```

CardBlock.jsx

```

import React from 'react';
import PropTypes from "prop-types";
import API from "../api/index";
import regeneratorRuntime from "regenerator-runtime";
import { apiPrefix } from "../../etc/config.json";
import constants from "./constants.js";
import CardCreator from "./CardCreator.jsx";
import dateFormat from 'dateformat';

dateFormat.i18n = {
    dayNames: [
        'ВС', 'ПН', 'ВТ', 'СР', 'ЧТ', 'ПТ', 'СБ',
        'Воскресенье', 'Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница', 'Суббота'
    ],
    monthNames: [
        'Янв', 'Фев', 'Мар', 'Апр', 'Май', 'Июн', 'Июл', 'Авг', 'Сент', 'Окт', 'Нояб', 'Дек',
        'Января', 'Февраля', 'Марта', 'Апреля', 'Мая', 'Июня', 'Июля', 'Августа', 'Сентября',
        'Октября', 'Ноября', 'Декабря'
    ],
    timeNames: [
        'a', 'p', 'ам', 'pm', 'A', 'P', 'AM', 'PM'
    ]
};

class CardBlock extends React.Component {
    constructor(props) {
        super(props);
    }
}

```

```

this.data = this.props.data;
this.id = this.data._id;
this.title = this.data.title;
this.text = this.data.text;
this.color = this.data.color;
this.createdAt = this.data.createdAt;
this.deleted = false;
this.handleDelete = this.handleDelete.bind(this);
this.handleEdit = this.handleEdit.bind(this);
this.handleEditor = this.handleEditor.bind(this);
this.editMode = false;
}

handleDelete() {
  this.deleted = true;
  this.callback(`DELETE/${this.id}`);
  this.forceUpdate();
}

handleEdit() {
  this.editMode = true;
  console.log(this.editMode);
  this.forceUpdate();
}

handleEditor(value) {
  this.editMode = false;
  this.callback(`EDIT/${this.id}`);
  this.forceUpdate();
}

componentDidUpdate(prevProps) {
  if (this.props.data._id != this.id || this.props.editMode != this.editMode ||
  this.props.title != this.title || this.props.text != this.text || this.props.color != this.color) {
    this.data = this.props.data;
    this.id = this.props.data._id;
    this.title = this.props.data.title;
    this.text = this.props.data.text;
    this.color = this.props.data.color;
    this.createdAt = this.props.data.createdAt;
    this.deleted = this.props.data.deleted;
    this.editMode = this.props.editMode;
  }
}

render() {

  let cardClass = " mt-4 mb-4"
  if (this.editMode)
    cardClass = "card bg-warning" + cardClass;
  else
    cardClass = "card bg-light" + cardClass;

  console.log(cardClass);

  let card = (
    <div className={cardClass} style={{"maxWidth": "600px"}}>
      <div className="card-header">{this.color}</div>
      <div className="card-body">
        <h5 className="card-title">{this.title}</h5>
        <p className="card-text">{this.text}</p>
        <p className="card-text small" style={{color: "grey"}}>{dateFormat(this.createdAt, "dddd, d mmmm / HH:MM")}</p>
        <button className="btn btn-danger" id="delete" onClick={this.handleDelete}>Удалить</button>
        <button className="btn btn-link" id="edit" onClick={this.handleEdit}>Редактировать</button>
      </div>
    </div>
  );
}

return (

```

```

        <div>
            {!this.deleted ? card : null}
            {this.editMode ? <div><h6 style={{"color" : "#b3b4b5"}}>РЕДАКТИРОВАНИЕ</h6><div
                className=" border rounded" style={{"maxWidth": "600px"}}><div className="p-3 border
                rounded"><CardCreator id={this.id} title={this.title} text={this.text} color={this.color}
                parentCallback = {this.handleEditor}/></div><br/></div> : null}
            </div>
        );
    }

    callback(value) {
        this.props.parentCallback(value);
    }
}

CardBlock.PropTypes = {
    data: PropTypes.string,
    id: PropTypes.string,
    title: PropTypes.string,
    text: PropTypes.string,
    color: PropTypes.string,
    createdAt: PropTypes.string,
    deleted: PropTypes.string
};

export default CardBlock;

```

Constants.js

```

export default({
    DEFAULT: 0,
    LOAD_CARDS_REQUEST: 1,
    SUCCESS_LOAD: 2,
    SEND_CARD_ERR: 3,
    TIMER_TIMEOUT: 4,
    CARD_DELETED: 5,
    CARD_EDIT_REQUEST: 6,
    SORT_REQUEST: 7
});

```

API/index.js

```

import axios from "axios";

import { apiPrefix } from '../../etc/config.json';

export default axios.create({
    baseURL: `${apiPrefix}`,
});

```

Сервер

App.js

```
import express from 'express';
import bodyParser from 'body-parser';
import cors from 'cors';

import { serverport } from '../etc/config.json';

import * as db from './utils/DataBaseUtils.js';
db.setUpConnection();

const app = express();

app.use(bodyParser.json());
app.use(cors({origin:true,credentials: true}));

app.get('/notes', (req, res) => {
    db.listNotes().then(data => res.send(data));
});

/*app.get('/', (req, res) => {
    res.send("admins don't drink chocolate / Oleg Kuznetsov, 2020");
});*/
app.get('/notes/sort/reversed', (req, res) => {
    db.listNotesReversed().then(data => res.send(data));
});

app.post('/notes', (req, res) => {
    db.createNote(req.body).then(data => res.send(data));
});

app.post('/notes/edit/:id', (req, res) => {
    db.editNote(req.params.id, req.body).then(data => res.send(data));
});

app.post('/notes/edit/withoutDateModify/:id', (req, res) => {
    db.editNoteWithoutDateModify(req.params.id, req.body).then(data => res.send(data));
});

app.delete('/notes/:id', (req, res) => {
    db.deleteNote(req.params.id).then(data => res.send(data));
});

app.get('/notes/filter/date/LowerBound/:date', (req, res) => {
    db.listNotes_dateLowerBound(req.params.date).then(data => res.send(data));
});

app.get('/notes/filter/date/UpperBound/:date', (req, res) => {
    db.listNotes_dateUpperBound(req.params.date).then(data => res.send(data));
});

app.get('/notes/filter/date/range/:startDate/:endDate', (req, res) => {
    db.listNotes_dateRange(req.params.startDate, req.params.endDate).then(data => res.send(data));
});

const server = app.listen(8080, () => {
    console.log('Server ready');
});
```

DatabaseUtils.js

```
import mongoose from "mongoose";
import '../models/Note';
import config from '../../etc/config.json';

const Note = mongoose.model('Note');

export function setUpConnection() {
    mongoose.connect(`mongodb://${config.db.host}:${config.db.port}/${config.db.name}`);
}
export function listNotes() {
    return Note.find().sort({createdAt: 'descending'});
}

export function listNotesReversed() {
    return Note.find().sort({createdAt: 'ascending'});
}

export function createNote(data) {
    const note = new Note({
        title: data.title,
        text: data.text,
        color: data.color,
        createdAt: new Date()
    });
    return note.save();
}

export function deleteNote(id) {
    return Note.findById(id).remove();
}

export function editNote(id, data) {
    return Note.findByIdAndUpdate(id, { title: data.title,
                                    text: data.text,
                                    color: data.color,
                                    createdAt: new Date()
                                });
}

export function editNoteWithoutDateModify(id, data) {
    return Note.findByIdAndUpdate(id, { title: data.title,
                                    text: data.text,
                                    color: data.color
                                });
}

export function listNotes_dateLowerBound(date) {
    return Note.find({createdAt: { $gte: date }});
}

export function listNotes_dateUpperBound(date) {
    return Note.find({createdAt: { $lte: date }});
}

export function listNotes_dateRange(startDate, endDate) {
    return Note.find(
        ({
            $and:
            [
                {createdAt: { $gte: startDate }},
                {createdAt: { $lte: endDate }}
            ]
        }));
}
```

Models/Note.js

```
import mongoose from "mongoose";

const Schema = mongoose.Schema;

const NoteSchema = new Schema({
  title      : { type: String },
  text       : { type: String, required: true },
  color      : { type: String },
  createdAt : { type: Date }
});

const Note = mongoose.model('Note', NoteSchema);
```

etc/Config.json

```
{
  "apiPrefix": "http://localhost:8080",
  "serverPort": "8080",
  "db": {
    "name": "notes",
    "host": "localhost",
    "port": 27017
  }
}
```