#### Министерство науки и высшего образования Российской Федераци­­и

#### Федеральное государственное бюджетное образовательное учреждение науки и высшего образования

#### «Владимирский государственный университет

#### имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

Лабораторная работа №8

по дисциплине

"Основы разработки веб приложений"

Тема: "Основы разработки с Headless CMS"

Выполнил:

ст. гр. ПРИ-117

В.Г. Хлызова

Принял:

Козаев А.Т.

Владимир, 2020 г.

## ЦЕЛЬ РАБОТЫ

Научится работать с Headless CMS. Создать приложение с использованием одной из Headless CMS.

ХОД РАБОТЫ

Устанавливаем Strapi с использованием npm: npm i strapi –g и генерируем проект: strapi new strapi.

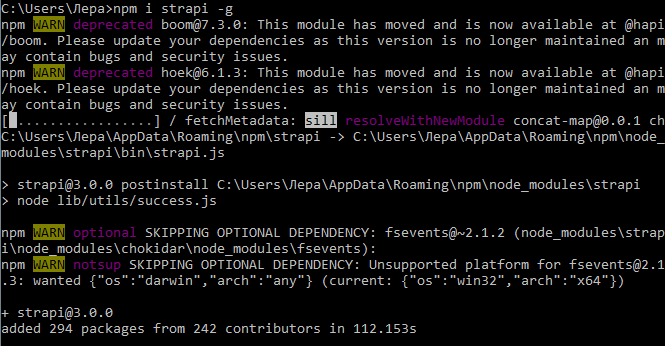


Рисунок 1. Установка strapi.

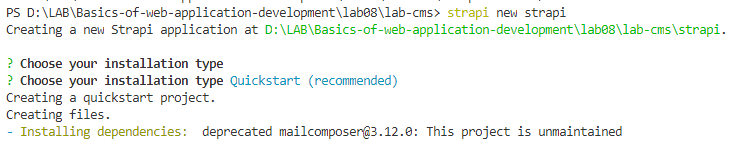


Рисунок 2. Генерируем проект.

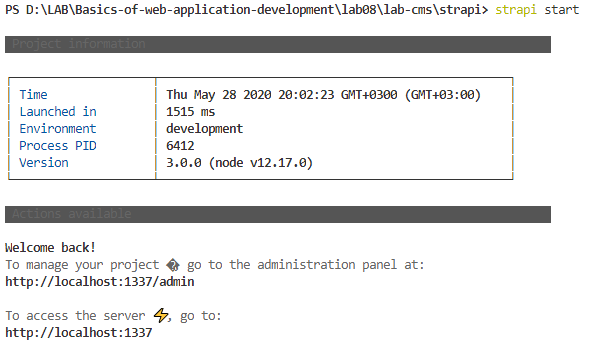


Рисунок 3. Запускаем сервер.

Открываем панель администрирования проекта, которая расположена по адресу <http://localhost:1337/admin> и создаем пользователя.

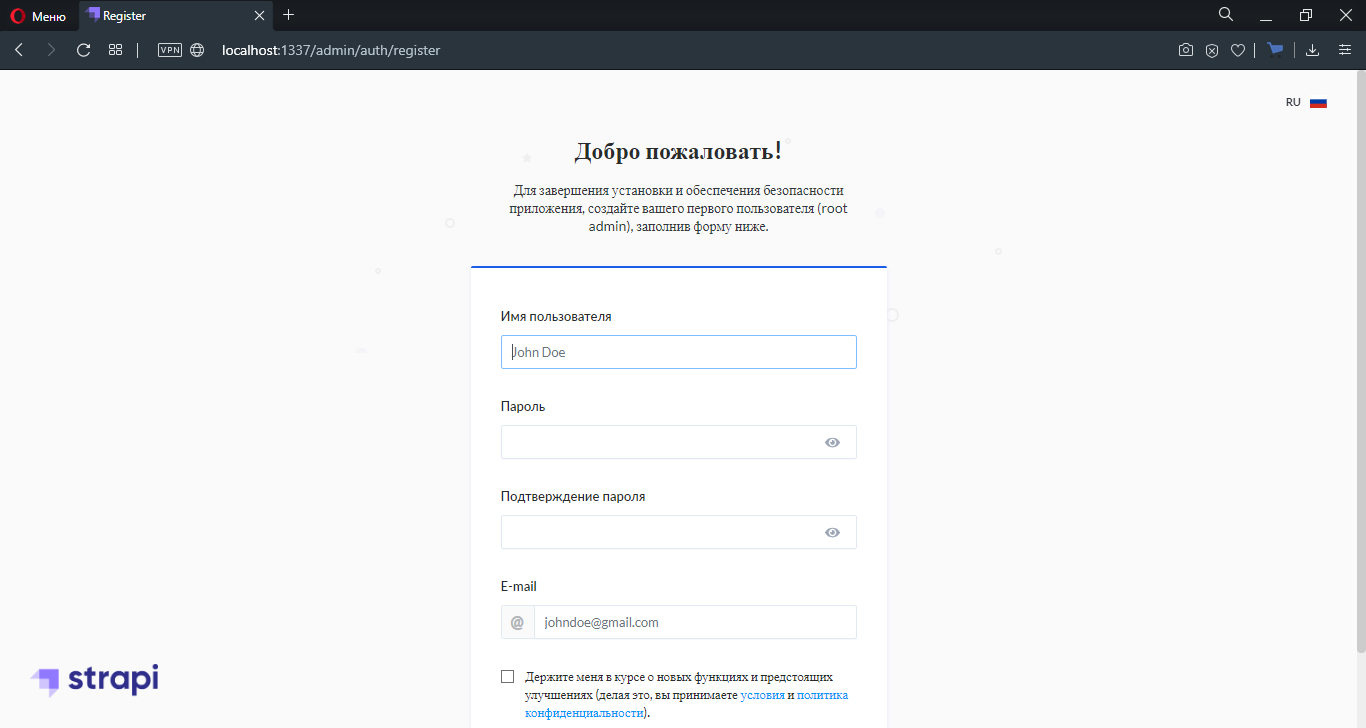


Рисунок 4. Панель администрирования.

Далее создаем три типа контента: post, author и category. В тип post добавляем поля с сылкой на типы author и category. Выбираем тип связи один ко многим.

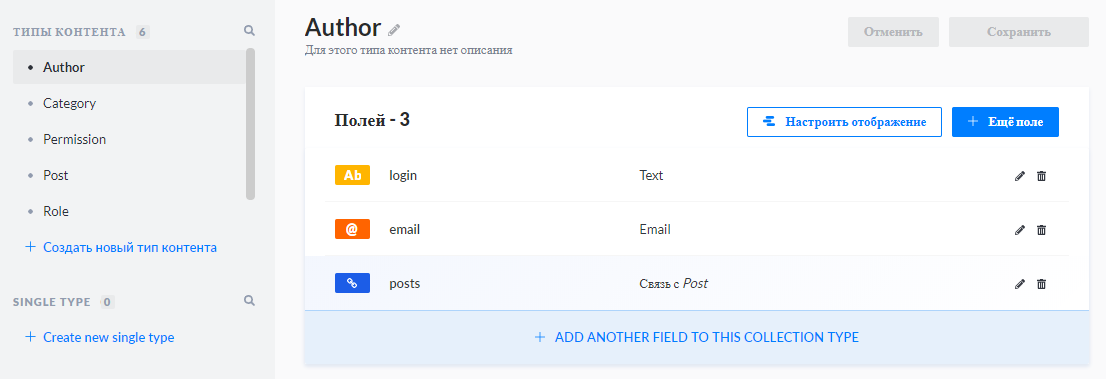


Рисунок 5. Тип контента Author.

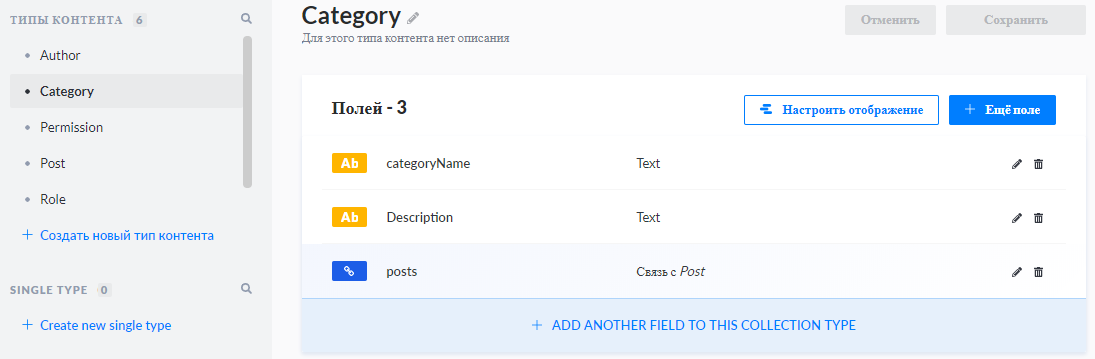


Рисунок 6. Тип контента Category.

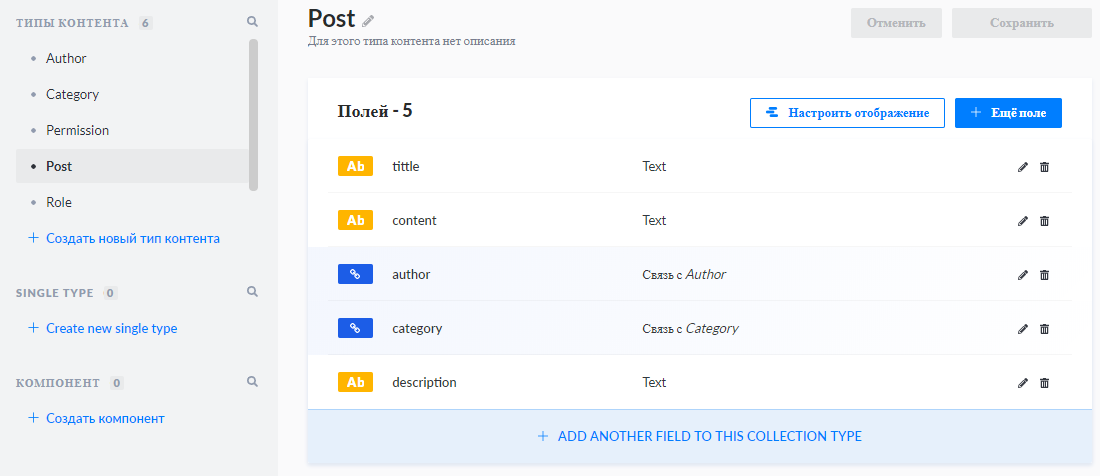


Рисунок 7. Тип контента Post.

Добавляем данные и добавляем открытый доступ к статьям.

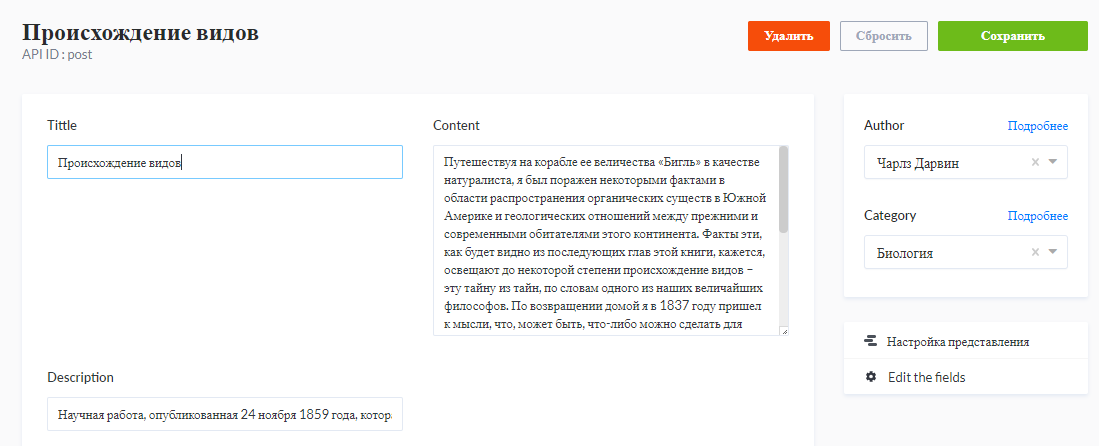


Рисунок 8. Добавляем данные.

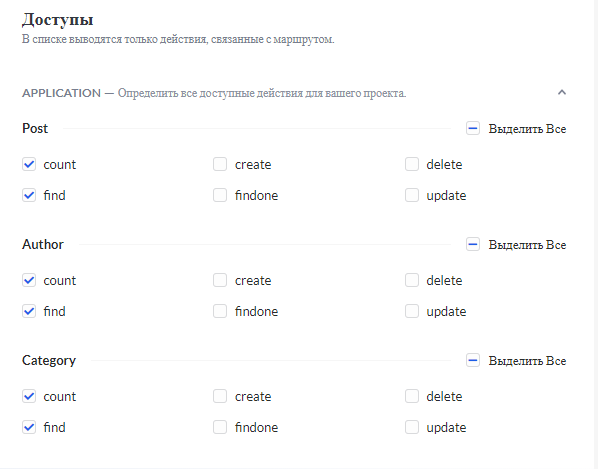


Рисунок 9. Открываем доступ.

Установим Gatsby CLI следующей командой: npm install --global gatsby-cli. Cоздаем новый блог: gatsby new blog и запускаем сервер: gatsby develop.

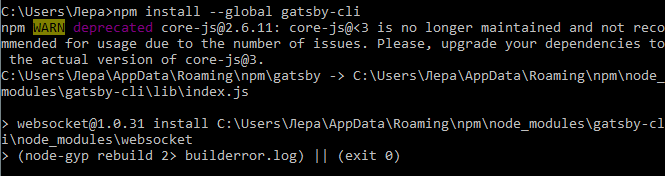


Рисунок 10. Установка Gatsby.

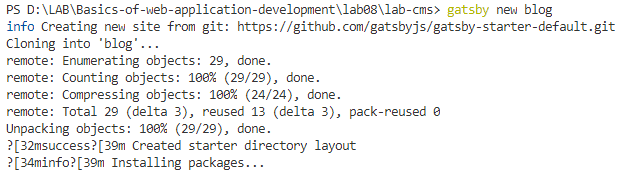


Рисунок 11. Создаем новый блог.

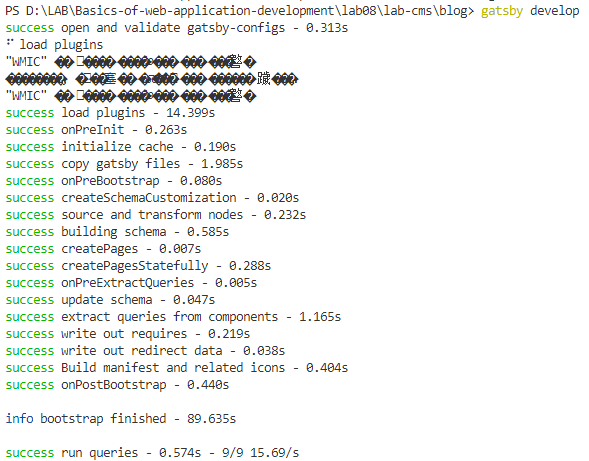


Рисунок 12. Запуск сервера.

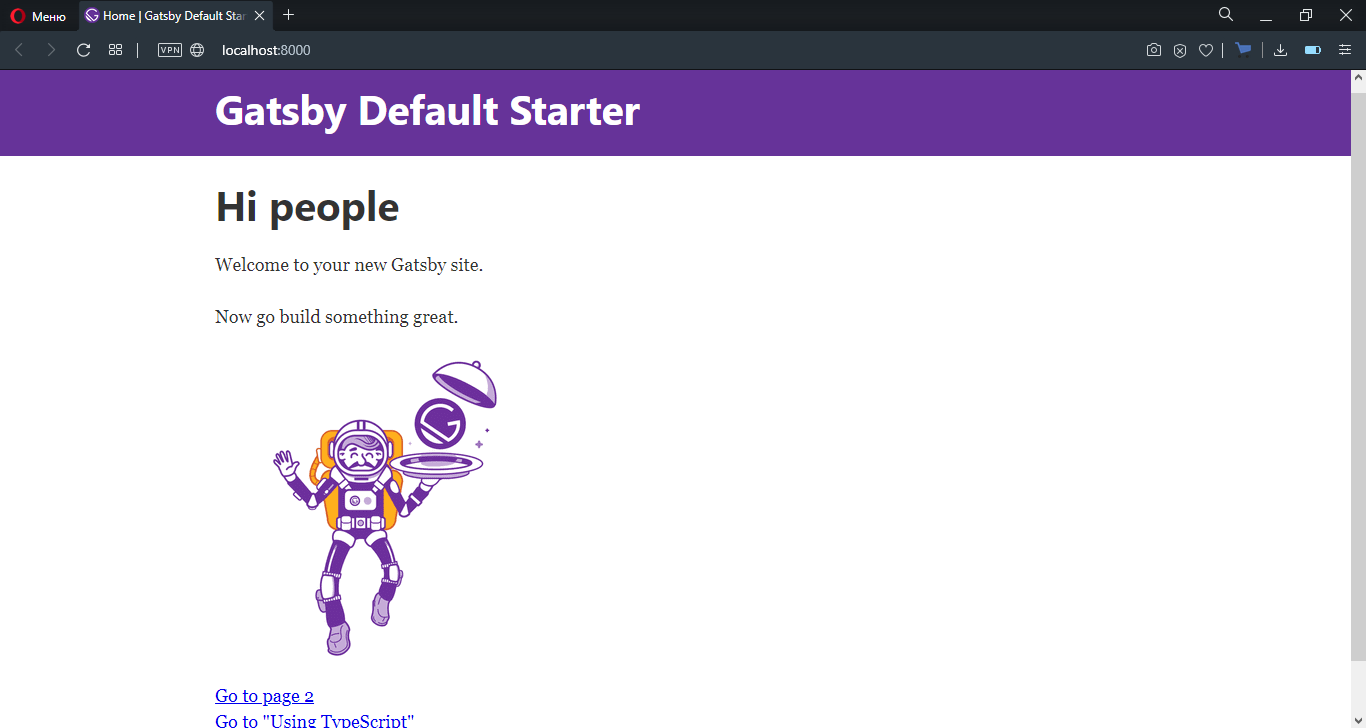


Рисунок 13. Результат запуска.

Устанавливаем плагин для того, чтобы подключить Gatsby к источнику данных: npm install --save gatsby-source-strapi (Рисунок 14). Далее настраиваем конфигурацию плагинов в файле gatsby-config.js:

module.exports = {

  plugins: [

    `gatsby-plugin-react-helmet`,

    {

      resolve: `gatsby-source-filesystem`,

      options: {

        name: `images`,

        path: `${\_\_dirname}/src/images`,

      },

    },

    {

      resolve: `gatsby-source-strapi`,

      options: {

        apiURL: `http://localhost:1337`,

        contentTypes: [

          `posts`,

          `authors`,

          `categories`

        ],

        queryLimit: 1000,

      },

    },

    `gatsby-transformer-sharp`,

    `gatsby-plugin-sharp`,

    {

      resolve: `gatsby-plugin-manifest`,

      options: {

        name: `gatsby-starter-default`,

        short\_name: `starter`,

        start\_url: `/`,

        background\_color: `#663399`,

        theme\_color: `#663399`,

        display: `minimal-ui`,

        icon: `src/images/gatsby-icon.png`,

      },

    },

    'gatsby-plugin-offline',

  ],

}

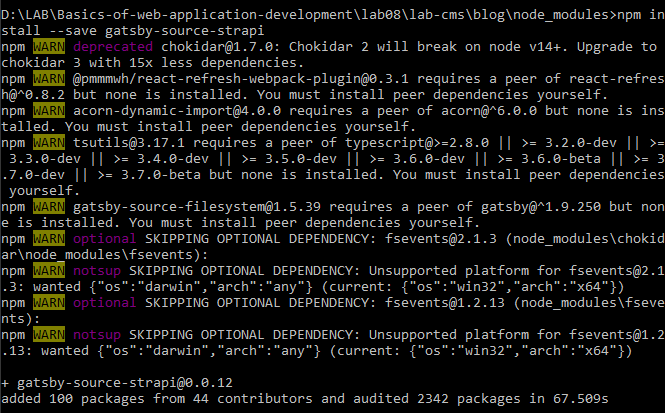


Рисунок 14. Установка плагина.

Выводим список всех статей на странице index.js:

import React from 'react'

import { Link, graphql } from "gatsby"

import Layout from "../components/layout"

const IndexPage = ({ data }) => (

  <Layout>

  <div>

    <p>

      <h3>Лабораторная работа №8.</h3>

      <h3>Основы разработки с Headless CMS.</h3>

    </p>

    <ul>

      {data.allStrapiPosts.edges.map(document => (

        <li key={document.node.id}>

          <font color="yellowgreen">

            <h4><Link to={`/${document.node.id}`}>{document.node.tittle}</Link></h4>

            <h4>Автор - <Link to={`/Authors\_${document.node.author.id}`}> {document.node.author.login}</Link></h4>

            <h4>Категория - <Link to={`/Categories\_${document.node.category.id}`}> {document.node.category.categoryName}</Link></h4>

          </font>

          <p>{document.node.description}</p>

        </li>

      ))}

    </ul>

  </div>

  </Layout>

)

export default IndexPage

export const pageQuery = graphql`

  query IndexQuery {

    allStrapiPosts {

      edges {

        node {

          id

          tittle

          description

          author{

            id

            login

          }

          category{

            id

            categoryName

          }

        }

      }

    }

  }

`

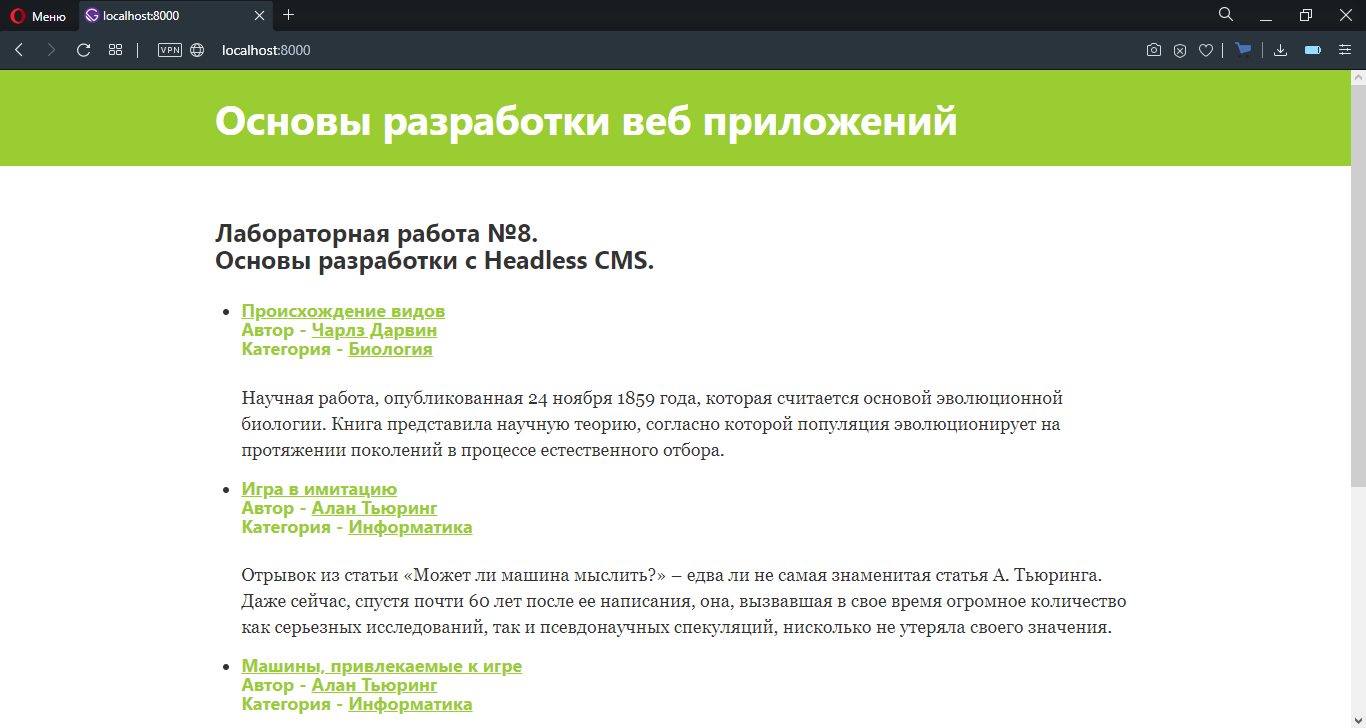


Рисунок 15. Вывод всех статей.

Вывод одной статьи:

import React from 'react'

import { Link, graphql } from 'gatsby'

import Layout from '../components/layout'

const PostTemplate = ({ data }) => (

  <Layout>

    <h1>{data.strapiPosts.tittle}</h1>

    <h3>Автор: {data.strapiPosts.author.login}</h3>

    <h3>Категория: {data.strapiPosts.category.categoryName}</h3>

    <p>{data.strapiPosts.content}</p>

  </Layout>

)

export default PostTemplate

export const query = graphql`

  query PostTemplate($id: String) {

    strapiPosts(id: {eq: $id}) {

      tittle

      content

      author {

        id

        login

      }

      category{

        id

        categoryName

      }

    }

  }

`

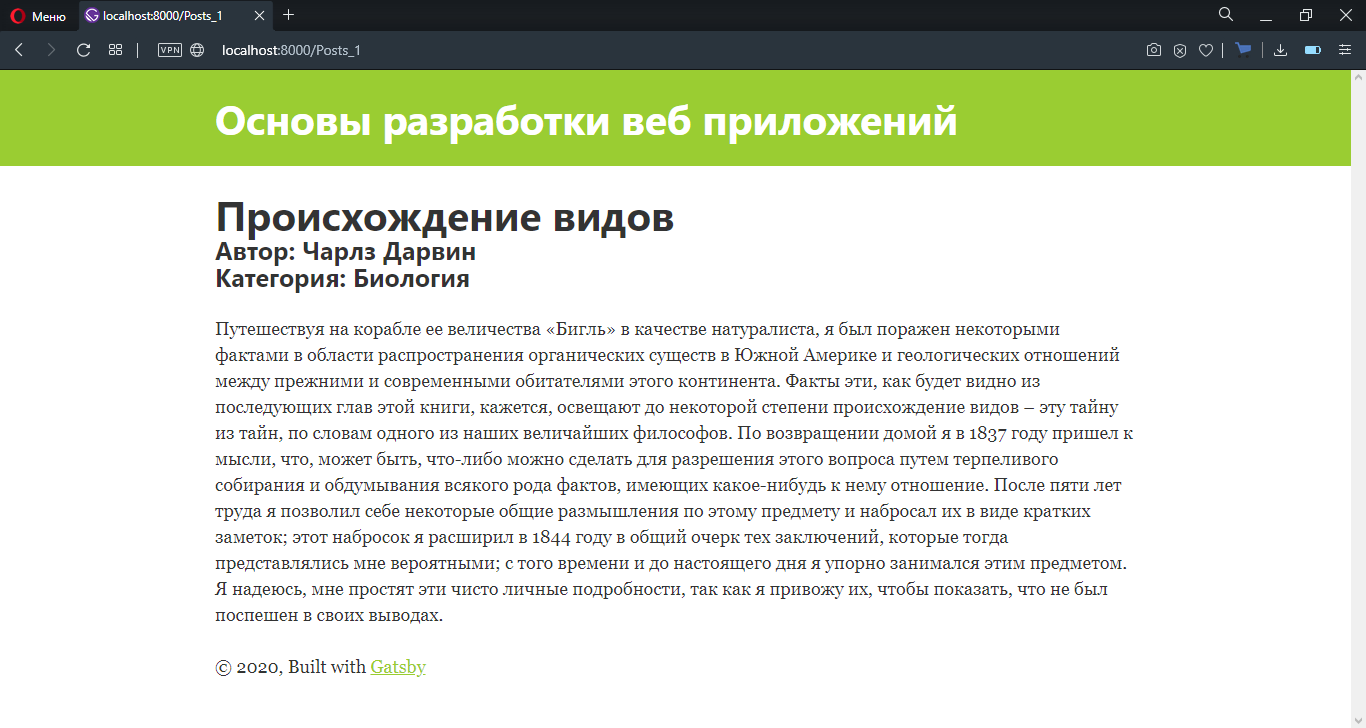


Рисунок 16. Вывод статьи.

Вывод информации об авторе и списка его статей:

import React from 'react'

import {Link, graphql } from 'gatsby'

import Layout from '../components/layout'

const AuthorTemplate = ({ data }) => (

    <Layout>

      <p><h1>{data.strapiAuthors.login}</h1>

      <h3>{data.strapiAuthors.email}</h3></p>

      <ul>

      {data.strapiAuthors.posts.map(document => (

        <li key={document.id}>

          <font color="yellowgreen">

            <h4><Link to={`/Posts\_${document.id}`}>{document.tittle}</Link></h4>

          </font>

          <p>{document.description}</p>

        </li>

      ))}

    </ul>

    </Layout>

  )

export default AuthorTemplate

export const query = graphql`

  query AuthorTemplate($id: String) {

    strapiAuthors(id: { eq: $id }) {

      id

      login

      email

      posts{

        id

        tittle

        description

      }

    }

  }

`

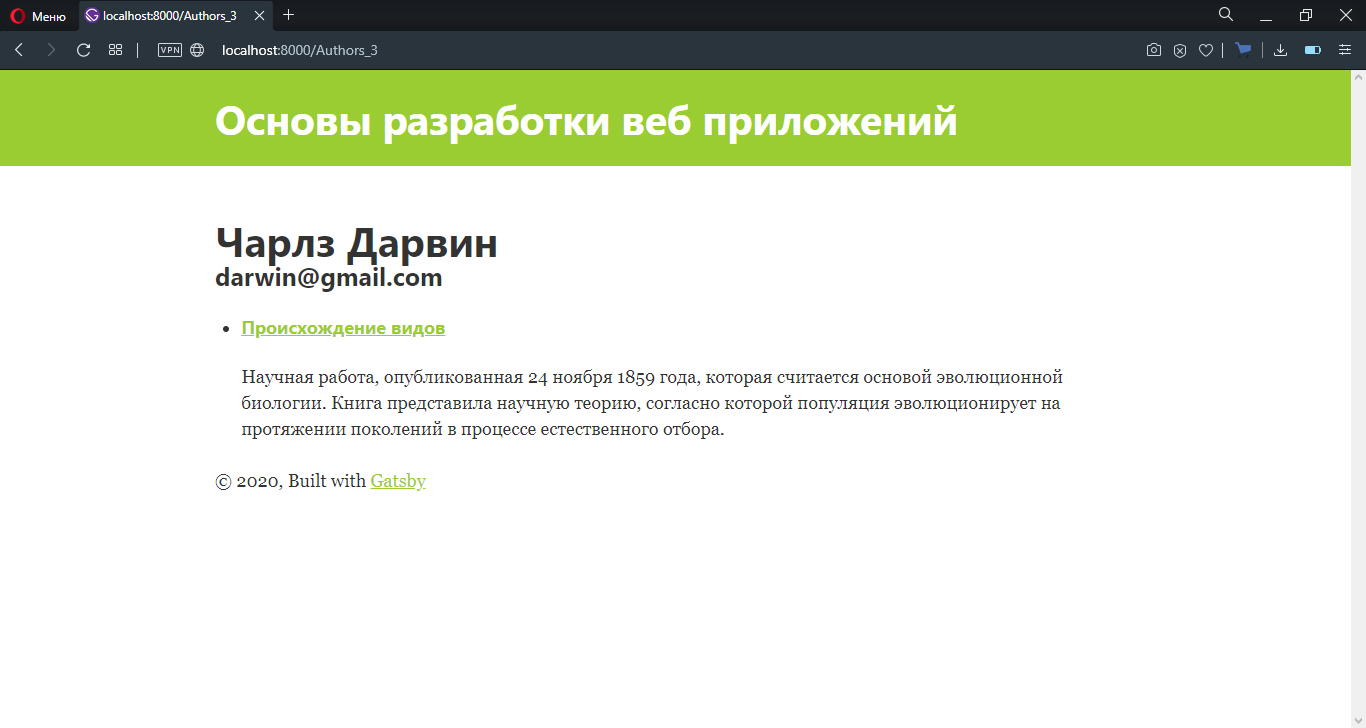


Рисунок 17. Страница автора.

Вывод всех статей определенной категории:

import React from 'react'

import { Link, graphql } from 'gatsby'

import Layout from '../components/layout'

const CategoryTemplate = ({ data }) => (

    <Layout>

      <p><h1>{data.strapiCategories.categoryName}</h1>

       <h3>{data.strapiCategories.Description}</h3></p>

       <ul>

      {data.strapiCategories.posts.map(document => (

        <li key={document.id}>

          <font color="yellowgreen">

            <h4><Link to={`/Posts\_${document.id}`}>{document.tittle}</Link></h4>

          </font>

          <p>{document.description}</p>

        </li>

      ))}

    </ul>

    </Layout>

  )

export default CategoryTemplate

export const query = graphql`

  query CategoryTemplate ($id: String) {

    strapiCategories(id: { eq: $id }) {

      id

      categoryName

      Description

      posts{

        id

        tittle

        description

      }

    }

  }

`

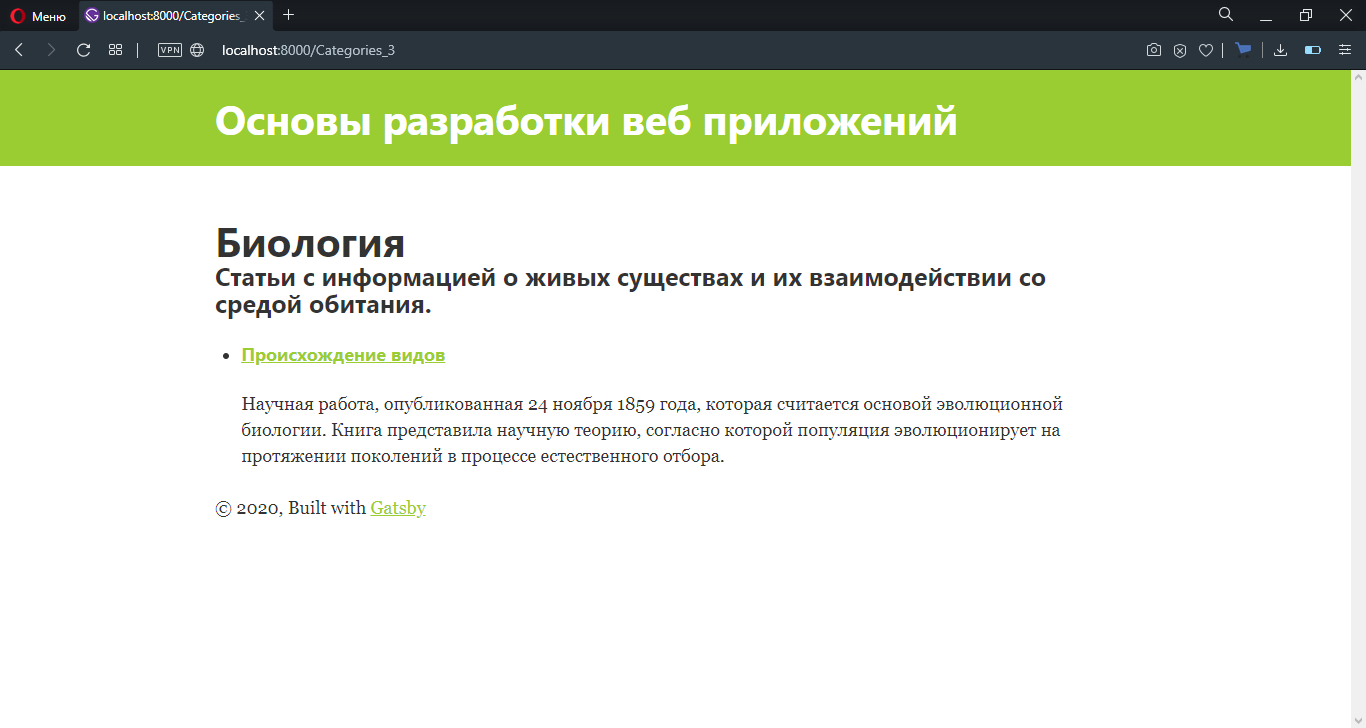


Рисунок 18. Вывод статей категории "Биология".

Для каждой сущности нужна собственная страницы. Генерация страниц для всех сущностей происходит в файле gatsby-node.js:

const path = require(`path`);

const makeRequest = (graphql, request) => new Promise((resolve, reject) => {

  resolve(

    graphql(request).then(result => {

      if (result.errors) {

        reject(result.errors)

      }

      return result;

    })

  )

});

exports.createPages = ({ boundActionCreators, graphql }) => {

  const { createPage } = boundActionCreators;

  const getPosts = makeRequest(graphql, `

    {

      allStrapiPosts {

        edges {

          node {

            id

          }

        }

      }

    }

    `).then(result => {

    result.data.allStrapiPosts.edges.forEach(({ node }) => {

      createPage({

        path: `/${node.id}`,

        component: path.resolve(`src/pages/post.js`),

        context: {

          id: node.id,

        },

      })

    })

  });

  const getAuthors = makeRequest(graphql, `

    {

      allStrapiAuthors {

        edges {

          node {

            id

          }

        }

      }

    }

    `).then(result => {

    result.data.allStrapiAuthors.edges.forEach(({ node }) => {

      createPage({

        path: `/${node.id}`,

        component: path.resolve(`src/pages/author.js`),

        context: {

          id: node.id,

        },

      })

    })

  });

  const getCategories = makeRequest(graphql, `

    {

      allStrapiCategories {

        edges {

          node {

            id

          }

        }

      }

    }

    `).then(result => {

    result.data.allStrapiCategories.edges.forEach(({ node }) => {

      createPage({

        path: `/${node.id}`,

        component: path.resolve(`src/pages/category.js`),

        context: {

          id: node.id,

        },

      })

    })

  });

return Promise.all([

    getPosts,

    getAuthors,

    getCategories

  ])

};

Ссылка на гит с приложением: <https://github.com/arranay/Basics-of-web-application-development>

ВЫВОД

В ходе лабораторной работы было создано приложение с использованием одной из Headless CMS - Strapi.