Development of a prototype web-based search system for driving school jobs in scientific and educational activities *

Hamraeva Anora Shamsiddinovna
Kazan Federal University
Kazan, Russia
anora.hamrayewa@gmail.com

Medvedeva Olga Anatolievna
Kazan Federal University
Kazan, Russia
OAMedvedeva@kpfu.ru

Mustafina Svetlana Anatol'evna
Bashkir State University
Ufa, Russia
mustafina_sa@mail.ru

Abstract

The article shows the relevance of developing a web-based driving school information system. The main characteristics of the site, the requirements for it, the tasks of its use in the educational process are determined. The circle of site users is indicated, a form for developing its information structure is proposed, additional site information services aimed at openness and improving the quality of education are discussed.

Introduction

Currently, cars have become publicly accessible vehicles, which entails a phenomenal increase in the relevance of driving schools. To automate the work of companies use information systems to facilitate the workflow of organizations. The relevance of such systems is determined by the presentation of large amounts of data in the form of structured and ordered information.

Information system - an organizationally ordered set of documents or arrays of documents and information technologies, including using computer technology and communications that implement information processes [1]. Modern information systems are impossible without the use of databases and DBMS, therefore the concept of an information system is close in meaning to the term "database system". Ideal for the enterprise is a single system that allows you to satisfy all the necessary needs of employees.

The purpose of the work is to develop a prototype of a web-based information system for accounting the work of a driving school, which allows automating the work of the enterprise and facilitating the interaction between students and school administrators.

Main part

In the work, an information system was developed to make the driving school management process more convenient for customers and staff. This information system will allow interested users to enroll in courses, find out where the branches of a driving school are, get information about fleets and leave feedback, and administrators can keep track of students in a driving school, as well as accept new applications for training. To create the project on the server side, we used: python / Django, django-rest-framework for providing api, on the client side, the React.js library was used to create the visual interface, as well as nodejs and the create-react-app utility to run development server for testing and debugging. PyCharm development environment was chosen as the implementation of the Driving Schools information system. Advantages of this environment: smart auto-completion, automatic code generation, automatic addition of necessary import operators, etc. PyCharm was chosen to implement the graphical interface. For convenient navigation between files during development, the data was placed in folders based on their purpose.

Frontend is the client side of the user interface to the firmware of the service. In figure 1.1 you can see a description of the functional user interface.

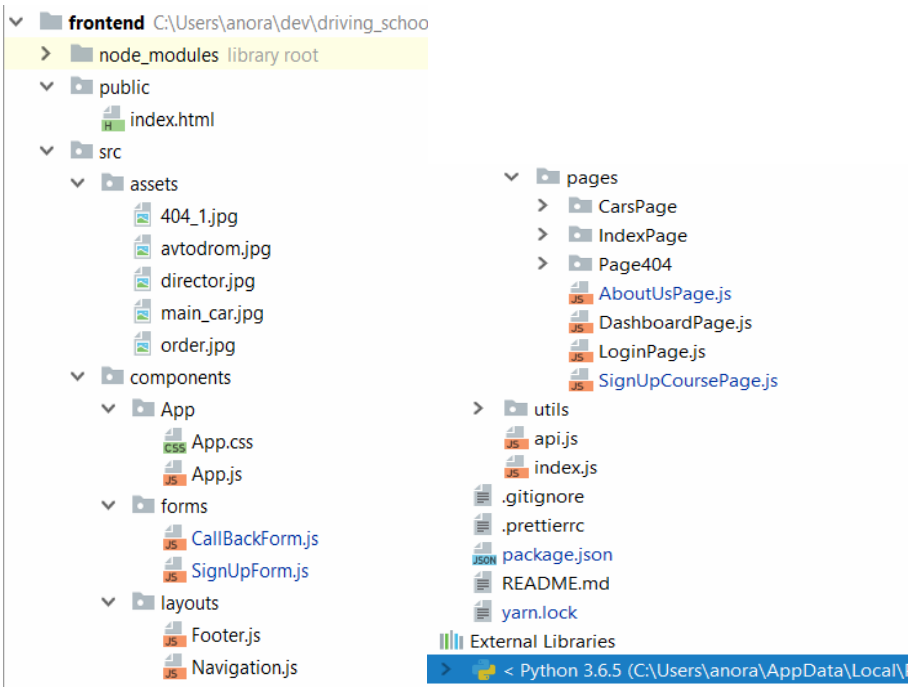

Figure 1.1. Frontend

The main code is located in the src folder, it contains the index.js file, figure 1.2. , which is the starting point of the entire application.

```
import React from "react";
import { render } from "react-dom";
import * as serviceWorker from "./utils/serviceWorker";
import { BrowserRouter } from "react-router-dom";
import { Route } from "react-router";

import App from "./components/App/App";
import "bootstrap/dist/css/bootstrap.min.css";

const rootElement = document.getElementById("root");
render(
  <BrowserRouter>
    <Route component={App} />
  </BrowserRouter>,
  rootElement
);

serviceWorker.register();
```

Figure 1.2. Index.js file

Here, the router is connected to navigate through the components of the page and the main component of the App is drawn onto the page, Figure 1.3. a), b).

```
import React, { Component } from "react";
import { Route, Switch } from "react-router";

import Navigation from "../layouts/Navigation";
import Footer from "../layouts/Footer";
import IndexPage from "../pages/IndexPage/IndexPage";
import AboutUsPage from "../pages/AboutUsPage";
import CarsPage from "../pages/CarsPage/CarsPage";
import Page404 from "../pages/Page404/Page404";
import SignUpCoursePage from "../pages/SignUpCoursePage";
import LoginPage from "../pages/LoginPage";
import DashboardPage from "../pages/DashboardPage";
import api from "../../api";
import setAuthHeaders from "../../utils/setAuthHeaders";
import "./App.css";

class App extends Component {
  state = {
    authorized: false,
    errors: "",
  };

  componentWillMount() {
    if (localStorage.DrivingToken) {
      setAuthHeaders(localStorage.DrivingToken);
      this.setState({ authorized: true });
    } else {
      this.setState({ authorized: false });
    }
  }

  logout = () => {
    this.setState({ authorized: false });
    setAuthHeaders();
    localStorage.removeItem("DrivingToken");
  };
```

Figure 1.3. Component App a)

```
  login = (login, password) =>
    api.auth
      .login({ username: login, password })
      .then(res => {
        localStorage.DrivingToken = res.access;
        setAuthHeaders(res.access);
        this.setState({ authorized: true, errors: "" });
        this.props.history.push("/dashboard");
      })
      .catch(err =>
        this.setState({ errors: "Authorization details are not correct" })
      );
  render() {
    const { authorized, errors } = this.state;
    return (
      <>
        <Navigation authorized={this.state.authorized} logout={this.logout} />
        <div className="content">
          <Switch>
            <Route path="/" exact component={IndexPage} />
            <Route path="/about" component={AboutUsPage} />
            <Route path="/cars" component={CarsPage} />
            <Route path="/order" component={SignUpCoursePage} />
            <Route
              path="/login"
              render={() => <LoginPage errors={errors} login={this.login} />}
            />
            <Route
              path="/dashboard"
              render={() => (authorized ? <DashboardPage /> : <Page404 />)}
            />
            <Route component={Page404} />
          </Switch>
        </div>
        <Footer />
      </>
    );
  }}
export default App;
```

Figure 1.3. Component App b)

```
<Route path="/about" component={AboutUsPage} />
```

Figure 1.4. Rendering the AboutUsPage Component

In figure 1.4. on the route about, the AboutUsPage component is drawn. The dashboard route is available only to authorized users, otherwise the program will display page 404. Authorization is based on JWT tokens, which are logged into localStorage and used to sign requests for confirmation of rights, this is implemented using the jios axios library, this http client adds the Authorization header. Also in the components folder there is a layouts folder that contains components that bump on each page, for example Navigation and Footer, the router at the address only changes the component located between them. The page directory contains page components such as LoginPage, IndexPage and others. An example of one of these pages is in Figure 1.5.

```jsx
import React from "react";

import boss from "../../assets/director.jpg";

const AboutUsPage = () => (
  <div className="container">
    <div className="row mb-3">
      <div className="col text-center">
        <h1>О нас</h1>
      </div>
    </div>
    <div className="row">
      <div className="col">
        <img src={boss} alt="boss" />
      </div>
      <div className="col">
        <h3>Our director: Anora Hamraeva</h3>
        <p>
          She has been driving a driving school for 6 years, she has won many awards,
            thanks to her thousands of people were able to get a driver's license!
        </p>
        <p>School address: Bolshaya Lubyanka, Moscow</p>
      </div>
    </div>
  </div>
);

export default AboutUsPage;
```

Figure 1.5. Page Component

To mark and position elements on the page, the bootstrap library [5] was used. Backend is a hardware-software part of a service. The root directory contains the following main files and directories:
• apps - Applications in the Django framework, reusable components
• conf - Directory supporting basic routing, and project settings
• manage.py - Squeak for project management, application of migrations, etc.
• db.sqlite3 - Database
• media - Contains media files
The apps contains a single application - school figure 1.7.

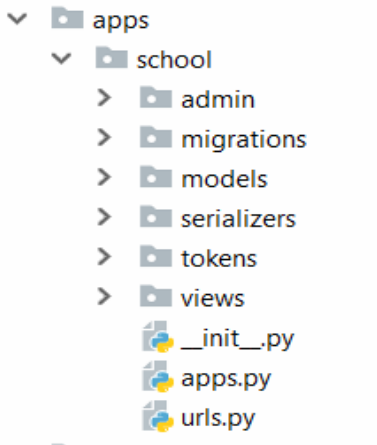

Figure 1.7. School

Migrations - migration files for interacting with the database.
Models - Models (tables) of the database. In figure 1.8.

```python
from django.db import models
from django.contrib.auth.models import User


class Car(models.Model):
    """Car model"""

    instructor = models.ForeignKey(User, on_delete=models.CASCADE)
    number = models.CharField(max_length=6)
    region = models.CharField(max_length=10)
    model = models.CharField(max_length=30)
    color = models.CharField(max_length=20, null=True)
    car_img = models.ImageField(null=True)

    class Meta:
        """Meta"""

        verbose_name = 'Машина'
        verbose_name_plural = 'Машины'
```

Figure 1.8. Database Models (Tables)

Serializers contains classes for serializing records from a database in json, for subsequent submission.

```python
from rest_framework_simplejwt.serializers import TokenObtainPairSerializer
from rest_framework_simplejwt.views import TokenObtainPairView


class ProjectTokenObtainPairSerializer(TokenObtainPairSerializer):
    """Token with addition keys"""

    def update(self, instance, validated_data):
        pass

    def create(self, validated_data):
        pass

    @classmethod
    def get_token(cls, user):
        token = super().get_token(user)

        token['username'] = user.username
        token['email'] = user.email

        return token


class ProjectTokenObtainPairView(TokenObtainPairView):
    """Custom token obtain view"""

    serializer_class = ProjectTokenObtainPairSerializer
```

Figure 1.9.Token

In figure 1.9. the token from the djangorestframework_simplejwt library is redefined. Email address and username are added. In figure 1.10. by post request to the server from the client part at the address orders. The OrderAPIView

view is called. Where is the entry in the orders table about the enrollment of a new student created? The name, category and instructor number are transmitted in the post request.

```python
from django.contrib.auth.models import User
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status

from ..models import Order


class OrderAPIView(APIView):
    """Order create"""

    def post(self, request):
        data = request.data
        print(data)
        if data.get('name') or data.get('categoryId') or data.get('instructorId'):
            Order.objects.create(
                name=data.get('name'),
                category=int(data.get('categoryId')),
                instructor=User.objects.get(pk=int(data.get('instructorId')))
            )
            return Response({'status': 'ok'})
        else:
            return Response(status=status.HTTP_400_BAD_REQUEST)
```

Figure 1.10. client part at orders

System Information Storage Design. To store the necessary information in the system, the SQLite database [4] was used. The system consists of the following tables (figure 1.11):

• User profile - name, username and password;
• Car - car model, instructor and color;
• Application for training - name, category of rights and instructor;
• Application for consultation - phone number and name;
• Reviews - text, username.

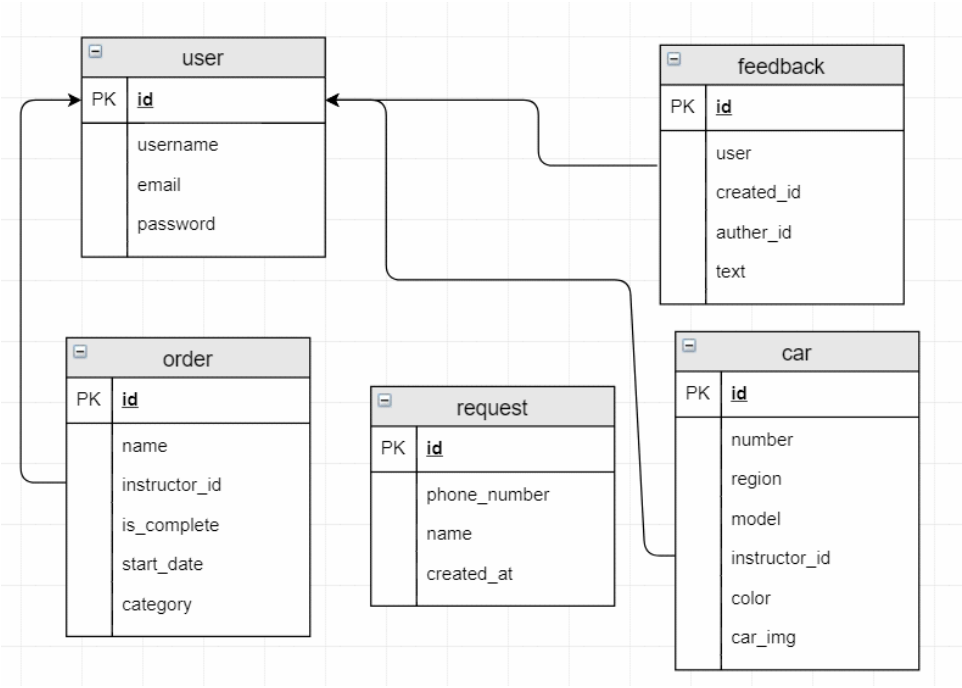

Figure 1.11. ER database diagram.

During testing, the method of 50 test cases was applied, which checks the operation of the site. Its essence is that it checks the operation of the main functions of the site and its correct display in all current browsers.

Conclusion

Information systems allow you to automate the work of companies, so their development is of high practical importance in the modern world. In order to speed up and increase the efficiency of the consultation process and application for training, the task was to develop a new information system for use by administrators and a user of a driving school. In the course of the work, an information system for a driving school was developed. This system will automate the work of a driving school and facilitate the process of interaction between the user and the administrator of the driving school. In the course of the work, a study of the subject area was carried out, the actors were identified, a graphical interface was designed and a prototype of the information system was developed.

Acknowledgements

The authors express their gratitude and appreciation to the organizers of the conference of the Ural Federal University and Innopolis University for their help and support in preparing the article.

References

1. Architecture and design of software systems: monograph / S.V. Nazarov. - 2nd ed., Revised. and add. - M.: INFRA-M, 2018. - 374 p.
2. Djangoproject Guide [Electronic Resource] - Access Mode: URL: https://www.djangoproject.com/
3. Python Guide [Electronic resource] - Access mode: URL: https://www.python.org/
4. Gagarina L.G. Introduction to software architecture: textbook. allowance / L.G. Gagarina, A.R. Fedorov, P.A. Fedorov. - M.: FORUM: INFRA-M, 2017. - 320 p.
5. GOST 19.201-1978. Technical task. Requirements for the content and design.; Enter 01-01-1990. - M.: Publishing house of standards, 1989. - 13 p.
6. GOST 34.601-1990. Automated systems. Stages of creation.; Enter 01-01-1992. - M.: Publishing house of standards, 1990. - 6 p.
7. Introduction to software engineering: Textbook / V.A. Antipov, A.A. Bubnov, A.N. Pylkin, V.K. Stolchnev. - M.: COURSE: INFRA-M, 2017. - 336 p.
8. Matorin S. I. Information systems: Training and practical guide / S.I. Matorin, O.A. Zimovets – Belgorod: Publishing House of Belgorod State University, 2012.
9. Official documentation about Getbootstrap [Electronic resource] - Access mode: URL: https://getbootstrap.com/ (03/15/2019).
10. Official JavaScript library documentation [Electronic resource] - Access mode: URL: https://reactjs.org/ (03/29/2019).
11. Andrianova A.A., Mukhtarova T.M., Rubtsova R.G. Laboratory workshop on the course 'Database Technologies': Textbook / A.A. Andrianova, T.M. Mukhtarova, R.G. Rubtsova. - Kazan: KFU, 2016. - 97 p.
12. SQlite Database Guide [Electronic resource] - Access mode: URL: https://www.sqlite.org/ (12.28.2018).