ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ассистент		А. Э. Зянчурин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

РАЗРАБОТКА ПРОТОТИПА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

по курсу: МЕТОДОЛОГИЯ ПРОГРАММНОЙ ИНЖЕНЕРИИ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4232М		В. Ф. Губайдулин
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

Цель работы:

Получение практических навыков, необходимых при разработке (конструировании) программного обеспечения, в соответствии со спецификацией требований.

Вариант 5:

Разработка программного обеспечения для автоматизации/информационной системы автобусного вокзала.

программному обеспечению).

Ход работы:

1. Разработка системы введётся на языках Python 3.10 с использованием фреймворка Django, а также TypeScript в связке с Angular 15. Разработку можно разделить на две части: backend и frontend.

2. Были описаны классы модели. При помощи встроенного в Django ORM есть возможно описанные модели мигрировать в базу данных.

Листинг 1 – Модели Django

```
from django.db import models


class DefaultVoyage(models.Model):
    voyage_number = models.CharField(max_length=255)
    time_departure = models.TimeField()
    end_time_departure = models.TimeField()
    days = models.PositiveSmallIntegerField()
    is_active = models.BooleanField(default=True)
    destination = models.CharField(max_length=255)

    def __str__(self):
        return self.voyage_number


class Bus(models.Model):
    licence_plate = models.CharField(max_length=255)
    sit_places = models.PositiveSmallIntegerField()
    is_broken = models.BooleanField(default=False)

    def __str__(self):
        return self.licence_plate


class Driver(models.Model):
    first_name = models.CharField(max_length=255)
    second_name = models.CharField(max_length=255)
    third_name = models.CharField(max_length=255)
    illness = models.BooleanField(default=False)
```

```
    hours_worked = models.PositiveSmallIntegerField()

    def __str__(self):
        return self.second_name


class Voyage(models.Model):
    voyage_number = models.ForeignKey('DefaultVoyage', on_delete=models.PROTECT,
null=False)
    date_departure = models.DateField()
    bus_id = models.ForeignKey('Bus', on_delete=models.PROTECT, null=True)
    driver_id = models.ForeignKey('Driver', on_delete=models.PROTECT, null=True)
    available_tickets = models.PositiveSmallIntegerField()

    def __str__(self):
        return str(self.voyage_number)
```

3. Были описаны сериализаторы для описанных ранее моделей. Сериализаторы необходимы для работы с данными, которые можно получить, используя запрос к базе через модели.

Листинг 2 – Сериализаторы моделей

```
from rest_framework import serializers
from rest_framework_simplejwt.serializers import TokenObtainPairSerializer

from .models import DefaultVoyage, Bus, Driver, Voyage


class CustomTokenObtainPairSerializer(TokenObtainPairSerializer):
    def validate(self, attrs):
        data = super().validate(attrs)
        groups = self.user.groups.values_list('name', flat=True)
        data['groups'] = groups
        return data


class VoyageSerializer(serializers.ModelSerializer):
    voyage_number                                                                =
serializers.PrimaryKeyRelatedField(queryset=DefaultVoyage.objects.all())
    driver_id                                                                    =
serializers.PrimaryKeyRelatedField(queryset=Driver.objects.all())
    bus_id = serializers.PrimaryKeyRelatedField(queryset=Bus.objects.all())
    default_voyage = serializers.SerializerMethodField()
    bus = serializers.SerializerMethodField()
    driver = serializers.SerializerMethodField()

    def get_default_voyage(self, obj):
        dv = obj.voyage_number
        dv_data = DefaultVoyageSerializer(dv).data
        return dv_data

    def get_bus(self, obj):
        bus = obj.bus_id
        bus_data = BusSerializer(bus).data
        return bus_data

    def get_driver(self, obj):
```

```
        driver = obj.driver_id
        driver_data = DriverSerializer(driver).data
        return driver_data

    class Meta:
        model = Voyage
        fields = "__all__"


class DefaultVoyageSerializer(serializers.ModelSerializer):
    class Meta:
        model = DefaultVoyage
        fields = "__all__"


class BusSerializer(serializers.ModelSerializer):
    class Meta:
        model = Bus
        fields = "__all__"


class DriverSerializer(serializers.ModelSerializer):
    class Meta:
        model = Driver
        fields = "__all__"
```

4. Для обращения к моделям были описаны view для каждой модели с применением подходящего сериализатора. View необходимы для обозначения методов API, которые будут доступны клиенту.

Листинг 3 – API view

```
import base64
from datetime import datetime
import io

import qrcode
from django.http import JsonResponse
from rest_framework import mixins
from rest_framework.decorators import action
from rest_framework.permissions import IsAuthenticatedOrReadOnly
from rest_framework.response import Response
from rest_framework.viewsets import GenericViewSet

from .serializers import VoyageSerializer, DriverSerializer, BusSerializer
from .models import Voyage, Driver, Bus


class BusViewSet(mixins.CreateModelMixin,
                 mixins.RetrieveModelMixin,
                 mixins.UpdateModelMixin,
                 mixins.ListModelMixin,
                 mixins.DestroyModelMixin,
                 GenericViewSet):
    serializer_class = BusSerializer
    permission_classes = (IsAuthenticatedOrReadOnly,)

    def get_queryset(self):
```

```
        pk = self.kwargs.get("pk")
        if not pk:
            return Bus.objects.all()
        return Bus.objects.filter(pk=pk)


class DriverViewSet(mixins.CreateModelMixin,
                    mixins.RetrieveModelMixin,
                    mixins.UpdateModelMixin,
                    mixins.ListModelMixin,
                    mixins.DestroyModelMixin,
                    GenericViewSet):
    serializer_class = DriverSerializer
    permission_classes = (IsAuthenticatedOrReadOnly,)

    def get_queryset(self):
        pk = self.kwargs.get("pk")
        if not pk:
            return Driver.objects.all()
        return Driver.objects.filter(pk=pk)


class VoyageViewSet(mixins.CreateModelMixin,
                    mixins.RetrieveModelMixin,
                    mixins.UpdateModelMixin,
                    mixins.ListModelMixin,
                    mixins.DestroyModelMixin,
                    GenericViewSet):
    serializer_class = VoyageSerializer
    permission_classes = (IsAuthenticatedOrReadOnly,)

    def get_queryset(self):
        pk = self.kwargs.get("pk")
        if not pk:
            return Voyage.objects.all()
        return Voyage.objects.filter(pk=pk)

    @action(methods=['get'], detail=True)
    def by_time(self, requset, date):
        qs = Voyage.objects.filter(date_departure=datetime.strptime(date, '%d-%m-
%Y').date())
        return JsonResponse(VoyageSerializer(qs, many=True).data, safe=False)

    @action(methods=['get'], detail=True)
    def qr(self, requset, price):
        string = 'Pay some: {}'.format(str(price))
        img = qrcode.make(string)
        temp = io.BytesIO()
        img.save(temp)
        img_b64 = base64.b64encode(temp.getvalue()).decode('utf-8')
        data = {'img': img_b64}
        return Response(data)
```

5. Для обозначения url адресов, доступных для обращения к API, необходимо описать их в соответствующем файле urls.

Листинг 4 – Файл urls

```
from django.contrib import admin
from django.urls import path, include
from rest_framework import routers
from rest_framework_simplejwt.views import TokenObtainPairView, TokenRefreshView,
TokenVerifyView

from station import views
from station.views import VoyageViewSet, DriverViewSet, BusViewSet

router = routers.DefaultRouter()
router.register(r'voyage', VoyageViewSet, basename='voyage')
router.register(r'driver', DriverViewSet, basename='driver')
router.register(r'bus', BusViewSet, basename='bus')

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/v1/', include(router.urls)),
    path('api/token/', TokenObtainPairView.as_view(), name='token_obtain_pair'),
    path('api/token/refresh/',                        TokenRefreshView.as_view(),
name='token_refresh'),
    path('api/token/verify/', TokenVerifyView.as_view(), name='token_verify'),
    path('api/v1/voyages/by_time/<str:date>/',
views.VoyageViewSet.as_view({'get': 'by_time'}), name='by_time'),
    path('api/v1/voyages/qr/<str:price>/',   views.VoyageViewSet.as_view({'get':
'qr'}), name='qr'),
    path('api/v1/voyages/drivers/',          views.VoyageViewSet.as_view({'get':
'allowed_drivers'}), name='allowed_drivers')
]
```

6. На frontend части был реализован CrudService для обращения к разрабатываемому API. Код CrudService представлен в приложении А.

7. Также были реализованы основные компоненты приложения. Основными компонентами являются:

- TicketBuyerComponent – модуль для покупки билетов пользователем ticket_buyer_user. Логика и разметка компонента представлены в приложении Б и приложении В.
- ViewerComponent – модуль, реализующий табло. Логика и разметка компонента представлены в приложении Г и приложении Д.
- VoyageEditor – модуль редактирование рейсов. Логика и разметка компонента представлены в приложении Е и приложении Ж.

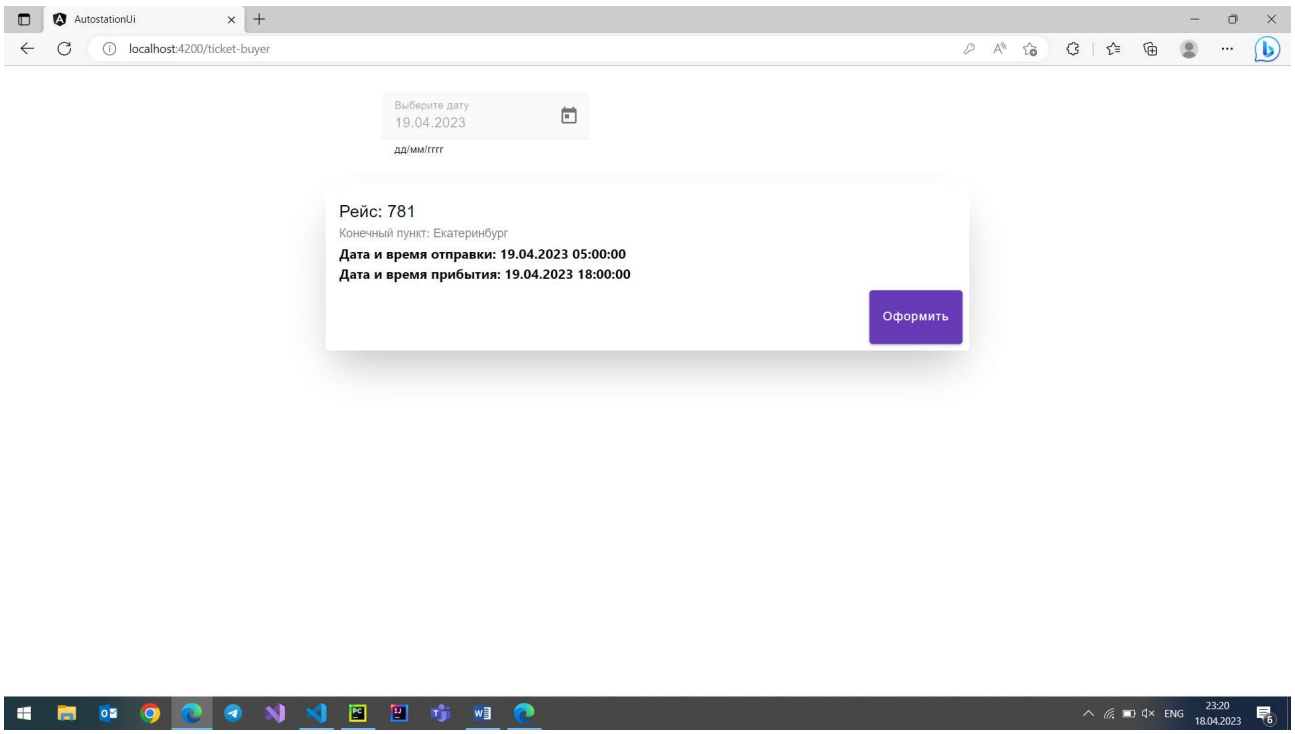

Рисунок 1 – Список рейсов для покупателя

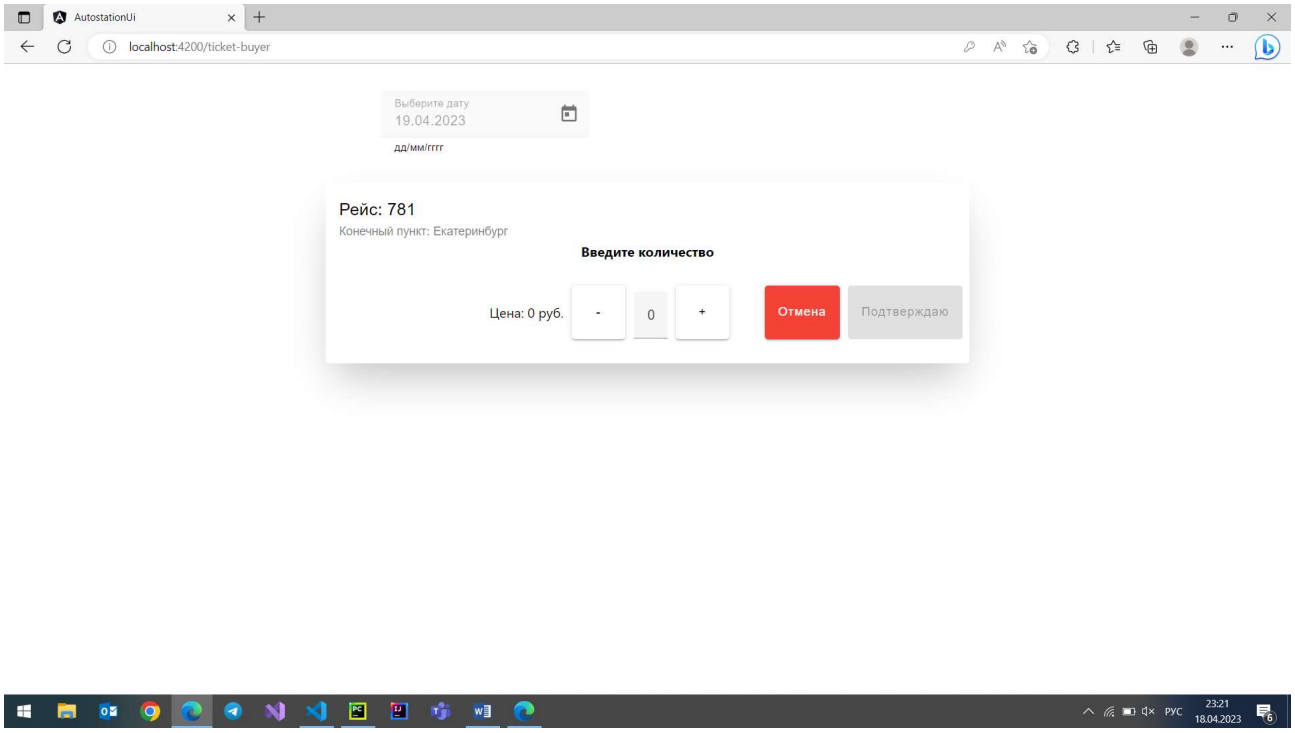

Рисунок 2 – Выбор количество билетов

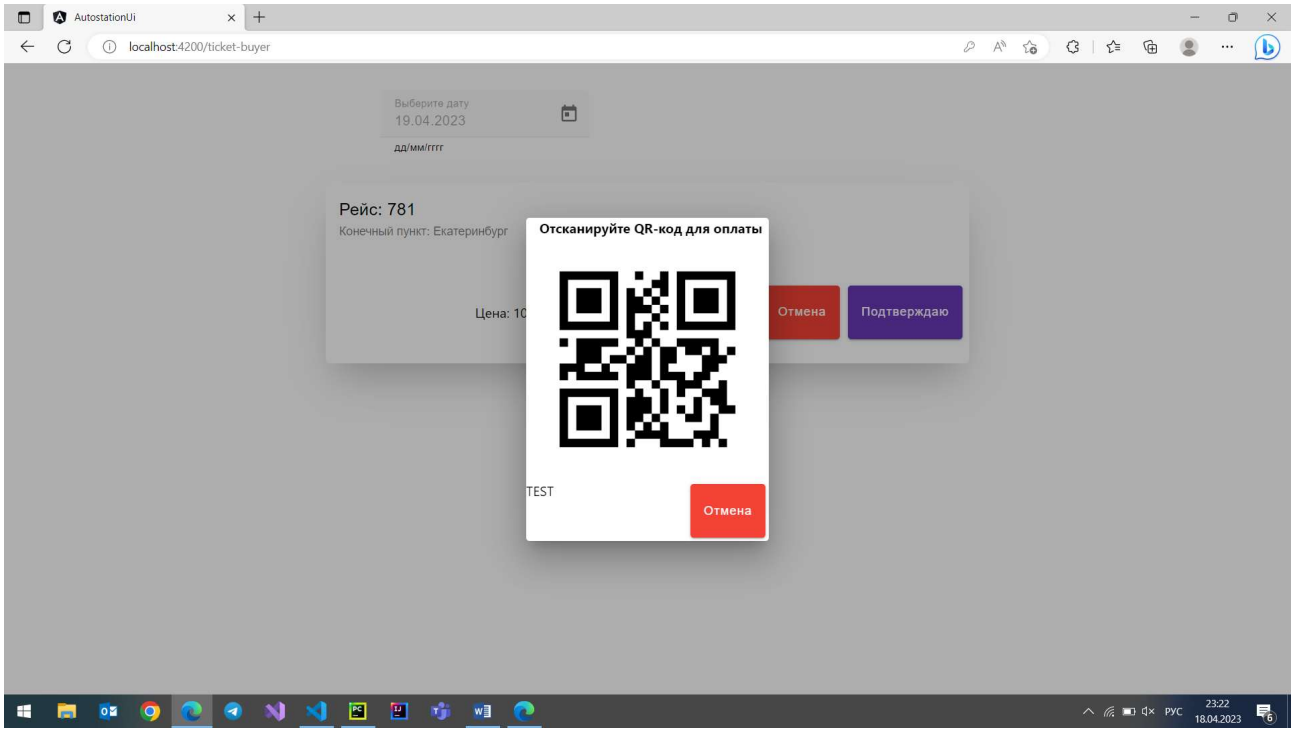

Рисунок 3 – Диалоговое окно с QR-кодом

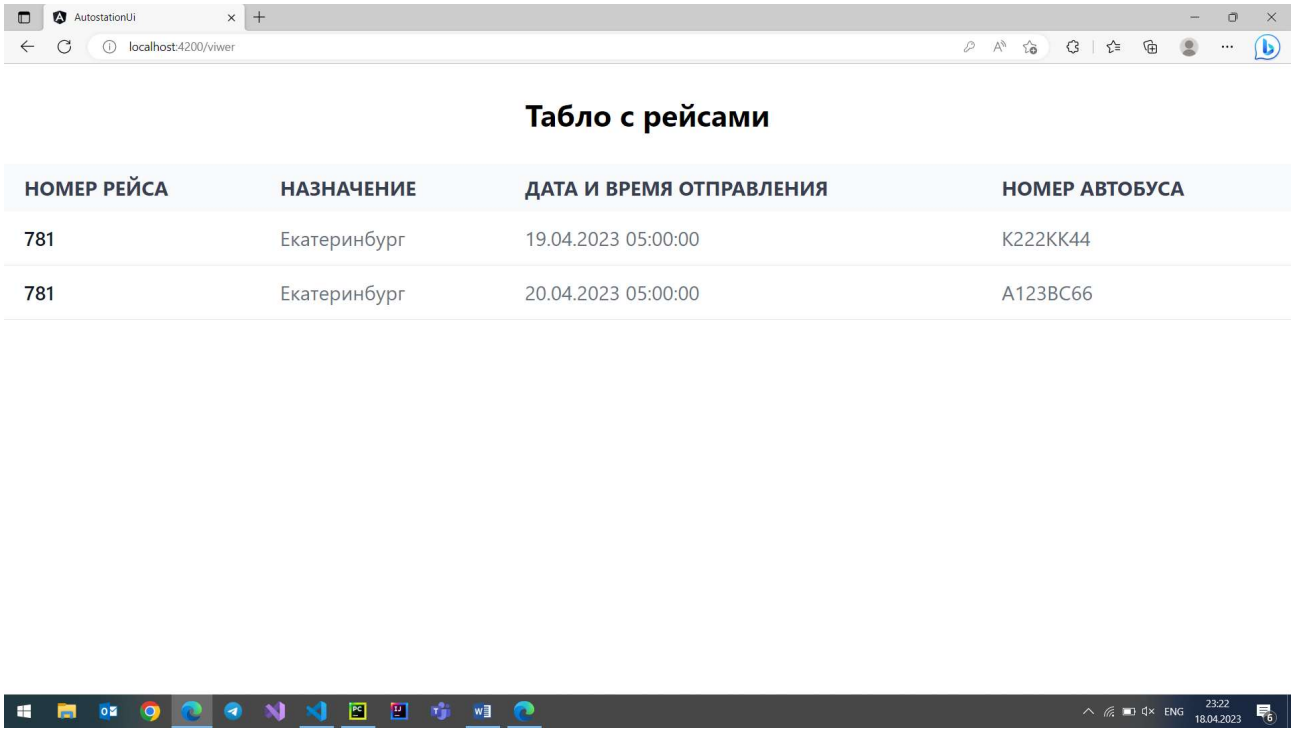

Рисунок 4 – Табло

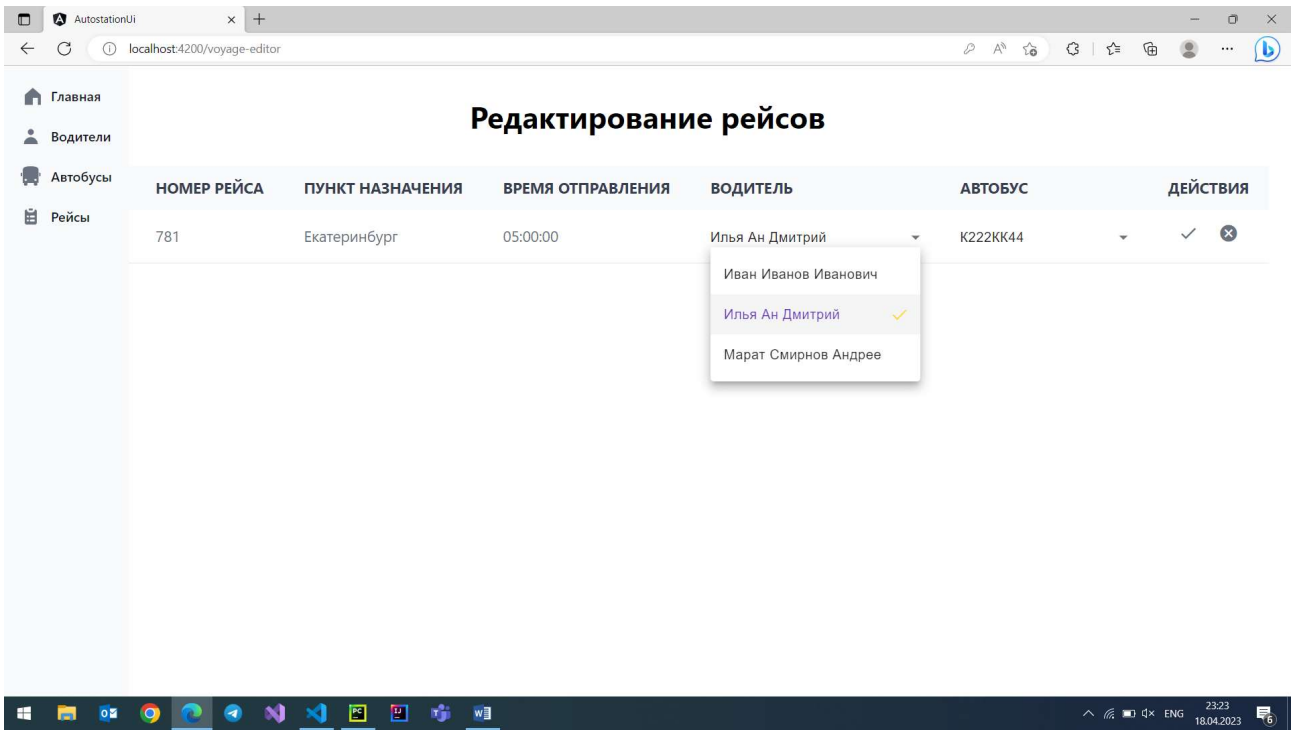

Рисунок 5 – Редактирование рейсов

Вывод:

Был реализован прототип системы для автовокзала с тремя основными модулями: покупка билетов, табло и АРМ оператора.

Приложение А. CrudService

```
import { HttpClient, HttpHeaders, HttpResponse } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { ErrorHandlerService } from '../error-handling/error-handler.service';
import { Token } from 'src/app/models/token';
import { catchError } from 'rxjs/operators';
import { HttpMethod } from 'src/app/utils/http-method';
import { Observable } from 'rxjs';
import { Img } from 'src/app/models/img';
import { LocalStorageConstants } from 'src/app/common/constants';

@Injectable({
  providedIn: 'root'
})
export class CrudService {

  commonOption: any;

  constructor(
    protected readonly httpClient: HttpClient,
    protected readonly errorHandlerService: ErrorHandlerService) {
    this.commonOption = {
      headers: new HttpHeaders({'Content-Type': 'application/json'})
    };
  }

  postToken(apiUrl: string, item: string, httpOptions: any): Observable<Token> {
    return this.httpClient
            .post<Token>(this.getAbsoluteUrl(apiUrl), item, httpOptions)
            .pipe(catchError(this.getErrorHandler('post', apiUrl).bind(this)));
  }

  getAll<T>(apiUrl: string): Observable<Array<T>> {
    return this.httpClient.get<Array<T>>(this.getAbsoluteUrl(apiUrl),
this.commonOption)
            .pipe(catchError(this.getErrorHandler<Array<T>>('get',
apiUrl).bind(this)))
  }

  getByDate<T>(apiUrl: string, date: string): Observable<Array<T>> {
    return this.httpClient.get<Array<T>>(this.getAbsoluteUrl(apiUrl, date),
this.commonOption)
            .pipe(catchError(this.getErrorHandler<Array<T>>('get',
apiUrl).bind(this)))
  }

  getQr(apiUrl: string, cost: string): Observable<Img> {
    return this.httpClient.get<Img>(this.getAbsoluteUrl(apiUrl, cost),
this.commonOption)
            .pipe(catchError(this.getErrorHandler<Img>('get',
apiUrl).bind(this)))
  }

  post<TIn, TOut>(apiUrl: string, item: TIn): Observable<TOut> {
    return this.httpClient.post<TIn>(this.getAbsoluteUrl(apiUrl), item,
this.getAccess())
              .pipe(catchError(this.getErrorHandler<TIn>('post',
apiUrl).bind(this)));
  }

  update<TIn, TOut>(apiUrl: string, id: string, item: TIn): Observable<TOut> {
```

```
    return this.httpClient.put<TIn>(this.getAbsoluteUrl(apiUrl, id), item,
this.getAccess())
            .pipe(catchError(this.getErrorHandler<TIn>('put',
apiUrl).bind(this)));
  }

  delete<T>(apiUrl: string, id: number | string): Observable<T> {
    return this.httpClient.delete<T>(this.getAbsoluteUrl(apiUrl, id),
this.getAccess())
            .pipe(catchError(this.getErrorHandler<T>('delete',
apiUrl).bind(this)));
  }

  getAccess(): any {
    let access = JSON.parse(localStorage.getItem(LocalStorageConstants.Token) as
string)['access'];
    return {
      headers: new HttpHeaders({'Content-Type': 'application/json',
    'Authorization' : 'Bearer ' + access})
    };
  }

  protected getErrorHandler<T>(method: HttpMethod, apiUrl: string, result?: T)
    : (error: any | HttpResponse<T>) => Observable<any> {
    return (error: any | HttpResponse<T>) => {
        if (error instanceof HttpResponse) {
            this.errorHandlerService.handle(error.status, method, apiUrl);
        }
        if (error != undefined) {
          this.errorHandlerService.handleError(error)
        }
        throw result || error;
    };
  }

  protected getAbsoluteUrl(apiUrl: string, id?: number|string): string {
    return `${apiUrl}${id ? '/' + id + '/' : '/'}`;
  }
}
```

Приложение Б. Логика TicketBuyerComponent

```
import { DatePipe } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { Observable, tap } from 'rxjs';
import { ApiPath } from 'src/app/common/constants';
import { Voyage } from 'src/app/models/voyage';
import { CrudService } from 'src/app/services/crud/crud.service';

@Component({
  selector: 'app-ticket-buyer',
  templateUrl: './ticket-buyer.component.html',
  styleUrls: ['./ticket-buyer.component.scss']
})
export class TicketBuyerComponent implements OnInit{

  isLoading: boolean = true;
  httpOptions: any;
  voyages$: Observable<Array<Voyage>>;
  today: Date = new Date();
  dt: Date = new Date();

  constructor(private crudService:  CrudService,
    private datePipe: DatePipe){
  }

  ngOnInit(): void {
    this.uploadData();
  }

  uploadData(): void {
    this.isLoading = true;
    this.voyages$ = this.crudService.getByDate<Voyage>(ApiPath.GetVoyagesByDate,
      this.datePipe.transform(this.dt, 'dd-MM-yyyy') as string).pipe(
      tap(() => this.isLoading = false)
    );
  }
}
```

Приложение В. Разметка TicketBuyerComponent

```
<mat-spinner *ngIf="isLoading" class="absolute top-1/2 left-1/2 transform mt-
24">
</mat-spinner>

<div class="ml-[450px] mt-8">
    <mat-form-field appearance="fill">
        <mat-label>Выберите дату</mat-label>
        <input matInput [matDatepicker]="picker" disabled [min]="today"
[(ngModel)]="dt" (dateChange)="uploadData()">
        <mat-hint>дд/мм/гггг</mat-hint>
        <mat-datepicker-toggle matIconSuffix [for]="picker"></mat-datepicker-
toggle>
        <mat-datepicker touchUi #picker disabled="false"></mat-datepicker>
    </mat-form-field>
</div>
<app-voyage-view *ngFor="let voyage of voyages$ | async"
    [voyage]="voyage">
</app-voyage-view>
```

Приложение Г. Логика ViewerComponent

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { Observable, tap } from 'rxjs';
import { ApiPath } from 'src/app/common/constants';
import { Voyage } from 'src/app/models/voyage';
import { CrudService } from 'src/app/services/crud/crud.service';

@Component({
  selector: 'app-viwer',
  templateUrl: './viwer.component.html',
  styleUrls: ['./viwer.component.scss']
})
export class ViwerComponent implements OnInit, OnDestroy {

  isLoading: boolean = true;
  voyages$: Observable<Voyage[]>
  now: Date = new Date();

  private timeOut: any
  private changeTime: number = 30;

  constructor(private crudService: CrudService) {

  }
  ngOnDestroy(): void {
    clearTimeout(this.timeOut);
  }

  ngOnInit(): void {
    this.infiniteLoopRecall(this.changeTime);
  }

  uploadData(): void {
    this.isLoading = true;
    this.voyages$ = this.crudService.getAll<Voyage>(ApiPath.GetAllVoyages).pipe(
      tap(() => this.isLoading = false)
    );
  }

  fullDate(voyage: Voyage): Date {
    return  new Date(voyage.date_departure.toString() + ' ' +
voyage.default_voyage.time_departure.toString())
  }

  private infiniteLoopRecall(delay: number, func?: Function): void {
    const toMs = delay * 1000;
    this.uploadData();
    this.timeOut = setTimeout(() => {
      this.infiniteLoopRecall(delay);
    }, toMs);
  }
}
```

Приложение Д. Разметка ViewerComponent

```
<mat-spinner *ngIf="isLoading" class="absolute top-1/2 left-1/2 transform mt-
24">
</mat-spinner>
<h1 class="text-4xl font-bold mb-10 mt-10 text-center">Табло с рейсами</h1>
<div class="relative overflow-x-auto">
    <table class="w-full text-sm text-left text-gray-500 dark:text-gray-400">
        <thead class="text-2xl text-gray-700 uppercase bg-gray-50 dark:bg-gray-
700 dark:text-gray-400">
            <tr>
                <th scope="col" class="px-6 py-3">
                    Номер рейса
                </th>
                <th scope="col" class="px-6 py-3">
                    Назначение
                </th>
                <th scope="col" class="px-6 py-3">
                    Дата и время отправления
                </th>
                <th scope="col" class="px-6 py-3">
                    Номер автобуса
                </th>
            </tr>
        </thead>
        <tbody>
            <tr *ngFor="let voyage of voyages$ | async" class="bg-white border-b
dark:bg-gray-800 dark:border-gray-700 text-2xl">
                <th *ngIf="fullDate(voyage) >= now" scope="row" class="px-6 py-4
font-medium text-gray-900 whitespace-nowrap dark:text-white">
                    {{voyage.default_voyage.voyage_number}}
                </th>
                <td *ngIf="fullDate(voyage) >= now" class="px-6 py-4">
                    {{voyage.default_voyage.destination}}
                </td>
                <td *ngIf="fullDate(voyage) >= now" class="px-6 py-4">
                    {{voyage.date_departure | date: 'dd.MM.yyyy'}}
{{voyage.default_voyage.time_departure}}
                </td>
                <td *ngIf="fullDate(voyage) >= now" class="px-6 py-4">
                    {{voyage.bus.licence_plate}}
                </td>
            </tr>
        </tbody>
    </table>
</div>
```

Приложение Е. Логика VoyageEditor

```
import { DatePipe } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import * as moment from 'moment';
import { Observable, tap } from 'rxjs';
import { ApiPath } from 'src/app/common/constants';
import { Bus } from 'src/app/models/bus';
import { Driver } from 'src/app/models/driver';
import { Voyage } from 'src/app/models/voyage';
import { CrudService } from 'src/app/services/crud/crud.service';
import { addDays } from 'src/app/utils/dt-helper';

@Component({
  selector: 'app-voyage-editor',
  templateUrl: './voyage-editor.component.html',
  styleUrls: ['./voyage-editor.component.scss']
})
export class VoyageEditorComponent implements OnInit {

  voyages$: Observable<Array<Voyage>>;
  isLoading: boolean = true;
  activated: number = 0;
  today: Date = new Date();
  dt: Date = new Date();
  workingHours: number = 60;

  allowDrivers: Array<Driver> = []
  allowBuses: Array<Bus> = []

  allVoyages: Array<Voyage> = []
  allDrivers: Array<Driver> = []
  allBuses: Array<Bus> = []

  newVoyage: Voyage;

  constructor(private crudService:  CrudService,
    private datePipe: DatePipe) {}

  ngOnInit(): void {
    this.getData();
  }

  uploadData(): void {
    this.isLoading = true;
    this.crudService.update<Voyage, Voyage>(ApiPath.GetAllVoyages,
this.newVoyage.id.toString(), this.newVoyage).subscribe(() => {
      this.getData();
    })
  }

  getData(): void {
    this.activated = 0;
    this.getAllVoyages();
    this.getAllBuses();
    this.getAllDriver();
    this.getVoyagesAsync();
  }

  getVoyagesAsync(): void {
    this.voyages$ = this.crudService.getByDate<Voyage>(ApiPath.GetVoyagesByDate,
      this.datePipe.transform(this.dt, 'dd-MM-yyyy') as string).pipe(
```

```
      tap(() => this.isLoading = false)
    );
  }

  prepareNewVoyage(voyage: Voyage, index: number) {
    this.activated = index + 1;
    this.newVoyage = voyage;
    this.getAllowedDriversAndBusesData(this.newVoyage);
  }

  getAllowedDriversAndBusesData(voyage: Voyage) {
    const startDt = moment(voyage.date_departure + ' ' +
voyage.default_voyage.time_departure);
    this.fillAllowBuses(startDt);
    this.fillAllowDrivers(startDt);
  }

  fillAllowBuses(startDt: moment.Moment) {
    this.allowBuses = []
    this.allBuses.forEach(x => {
      let canAllow = true;
      let voyages = this.allVoyages.filter(x => x.bus.id == x.id);
      voyages.forEach(y => {
        let endDt = moment(addDays(new Date(y.date_departure),
y.default_voyage.days) + ' ' + y.default_voyage.end_time_departure);
        if (endDt >= startDt) {
          canAllow = false;
        };
      });
      if (canAllow == true) this.allowBuses.push(x);
    });
  }

  fillAllowDrivers(startDt: moment.Moment) {
    this.allowDrivers = []
    this.allDrivers.forEach(x => {
      let canAllow = true;
      let voyages = this.allVoyages.filter(x => x.driver.id == x.id);
      voyages.forEach(y => {
        let endDt = moment(addDays(new Date(y.date_departure),
y.default_voyage.days) + ' ' + y.default_voyage.end_time_departure);
        if (endDt >= startDt) {
          canAllow = false;
        };
      });
      if (canAllow == true && x.hours_worked as number <= this.workingHours)
this.allowDrivers.push(x);
    });
  }

  getAllBuses() {
    this.isLoading = true;
    this.crudService.getAll<Bus>(ApiPath.Bus).subscribe(result => {
      this.allBuses = result;
      this.isLoading = false;
    });
  }

  getAllDriver() {
    this.isLoading = true;
    this.crudService.getAll<Driver>(ApiPath.Driver).subscribe(result => {
      this.allDrivers = result;
```

```
      this.isLoading = false;
    });
  }

  getAllVoyages() {
    this.isLoading = true;
    this.crudService.getAll<Voyage>(ApiPath.GetAllVoyages).subscribe(result => {
      this.allVoyages = result;
      this.isLoading = false;
    });
  }
}
```

Приложение Ж. Разметка VoyageEditorComponent

```
<mat-spinner *ngIf="isLoading" class="absolute top-1/2 left-1/2 transform mt-
24"></mat-spinner>
<app-nav-bar style="float: left;"></app-nav-bar>
<h1 class="text-4xl font-bold mb-10 mt-10 flex items-center justify-
center">Редактирование рейсов</h1>
<div class="ml-[450px] mt-8" *ngIf="activated==0">
    <mat-form-field appearance="fill">
        <mat-label>Выберите дату</mat-label>
        <input matInput [matDatepicker]="picker" disabled [min]="today"
[(ngModel)]="dt" (dateChange)="getData()">
        <mat-hint>дд/мм/гггг</mat-hint>
        <mat-datepicker-toggle matIconSuffix [for]="picker"></mat-datepicker-
toggle>
        <mat-datepicker touchUi #picker disabled="false"></mat-datepicker>
    </mat-form-field>
</div>
<div class="flex items-center justify-center">
    <div class="relative overflow-x-auto">
        <form>
            <table class="w-6/12 text-sm text-left text-gray-500 dark:text-gray-
400 whitespace-nowrap">
                <thead class="text-lg text-gray-700 uppercase bg-gray-50
dark:bg-gray-700 dark:text-gray-400">
                    <tr>
                        <th scope="col" class="px-6 py-3">ID рейса</th>
                        <th scope="col" class="px-6 py-3">Номер рейса</th>
                        <th scope="col" class="px-6 py-3">Пункт назначения</th>
                        <th scope="col" class="px-6 py-3">Время отправления</th>
                        <th scope="col" class="px-6 py-3">Водитель</th>
                        <th scope="col" class="px-6 py-3">Автобус</th>
                        <th scope="col" class="px-6 py-3">Действия</th>
                    </tr>
                </thead>
                <tbody>
                    <tr *ngFor="let voyage of voyages$ | async; let i=index"
class="bg-white border-b dark:bg-gray-800 dark:border-gray-700 text-lg">
                        <ng-container *ngIf="activated!=i+1">
                            <th scope="row" class="px-6 py-4 font-medium text-
gray-900 whitespace-nowrap dark:text-white">
                                {{voyage.id}}
                            </th>
                            <td class="px-6 py-4">

<label>{{voyage.default_voyage.voyage_number}}</label>
                            </td>
                            <td  class="px-6 py-4">

<label>{{voyage.default_voyage.destination}}</label>
                            </td>
                            <td  class="px-6 py-4">

<label>{{voyage.default_voyage.time_departure}}</label>
                            </td>
                            <td  class="px-6 py-4">
                                <label>{{voyage.driver.first_name}}
{{voyage.driver.second_name}} {{voyage.driver.third_name}}</label>
                            </td>
                            <td  class="px-6 py-4">
                                <label>{{voyage.bus.licence_plate}}</label>
                            </td>
```

```
                                <td  class="px-6 py-4">
                                    <div class="flex items-center justify-center">
                                        <button class="mx-3" *ngIf="activated==0"
(click)="prepareNewVoyage(voyage, i)"><mat-icon>edit</mat-icon></button>
                                    </div>
                                </td>
                            </ng-container>
                            <ng-container *ngIf="activated==i+1">
                                <th scope="row" class="px-6 py-4 font-medium text-
gray-900 whitespace-nowrap dark:text-white">
                                    {{voyage.id}}
                                </th>
                                <td class="px-6 py-4">

<label>{{voyage.default_voyage.voyage_number}}</label>
                                </td>
                                <td  class="px-6 py-4">

<label>{{voyage.default_voyage.destination}}</label>
                                </td>
                                <td  class="px-6 py-4">

<label>{{voyage.default_voyage.time_departure}}</label>
                                </td>
                                <td  class="px-6 py-4">
                                    <mat-select style="width: 250px"
[(ngModel)]="newVoyage.driver_id" [ngModelOptions]="{standalone: true}">
                                        <mat-option *ngFor="let driver of
allowDrivers" [value]="driver.id">
                                            {{driver.first_name}}
{{driver.second_name}} {{driver.third_name}}
                                        </mat-option>
                                    </mat-select>
                                </td>
                                <td  class="px-6 py-4">
                                    <mat-select style="width: 200px"
[(ngModel)]="newVoyage.bus_id" [ngModelOptions]="{standalone: true}">
                                        <mat-option *ngFor="let bus of allowBuses"
[value]="bus.id">
                                            {{bus.licence_plate}}
                                        </mat-option>
                                    </mat-select>
                                </td>
                                <td  class="px-6 py-4">
                                    <div class="flex items-center justify-center">
                                        <button class="mx-3" *ngIf="activated==i+1"
(click)="uploadData()"><mat-icon>check</mat-icon></button>
                                        <button class="mx-3" *ngIf="activated==i+1"
(click)="getData()"><mat-icon>cancel</mat-icon></button>
                                    </div>
                                </td>
                            </ng-container>
                        </tr>
                    </tbody>
                </table>
            </form>
        </div>
    </div>
```