**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра Вычислительной Техники**

**ОТЧЕТ**

**по лабораторной работе №3**

**по дисциплине «Программирование в среде DOTNet»**

**ТЕМА: Разработка программного интерфейса для взаимодействия с приложением**

|  |  |  |  |
| --- | --- | --- | --- |
| Студенты гр. 6307 | |  | Лазарев С. О. |
| Преподаватель | |  | Пешехонов К. А. |
|  |
|  |  |  |  |

Санкт-Петербург

2020

**Цели работы:**

Реализовать программный интерфейс для взаимодействия с приложением.

Эта и следующие работы будут выполнены на языке Dart.

APPCHANNEL:

|  |
| --- |
| import 'package:aqueduct/managed\_auth.dart'; |
|  | import 'package:car\_drivers/api/controllers/cars\_controller.dart'; |
|  | import 'package:car\_drivers/api/controllers/drivers\_controller.dart'; |
|  | import 'package:car\_drivers/api/controllers/random\_controller.dart'; |
|  | import 'package:car\_drivers/data/datasource/car\_datasource.dart'; |
|  | import 'package:car\_drivers/data/datasource/driver\_datasource.dart'; |
|  | import 'package:car\_drivers/data/repository/car\_repository\_impl.dart'; |
|  | import 'package:car\_drivers/data/repository/driver\_repository\_impl.dart'; |
|  | import 'package:car\_drivers/domain/repository/car\_repository.dart'; |
|  | import 'package:car\_drivers/domain/repository/driver\_repository.dart'; |
|  | import 'package:car\_drivers/domain/usecase/collect\_random\_pair\_usecase.dart'; |
|  | import 'package:car\_drivers/domain/usecase/create\_car\_usecase.dart'; |
|  | import 'package:car\_drivers/domain/usecase/create\_driver\_usecase.dart'; |
|  | import 'package:car\_drivers/domain/usecase/delete\_car\_usecase.dart'; |
|  | import 'package:car\_drivers/domain/usecase/delete\_driver\_usecase.dart'; |
|  | import 'package:car\_drivers/domain/usecase/get\_car\_usecase.dart'; |
|  | import 'package:car\_drivers/domain/usecase/get\_cars\_usecase.dart'; |
|  | import 'package:car\_drivers/domain/usecase/get\_driver\_usecase.dart'; |
|  | import 'package:car\_drivers/domain/usecase/get\_drivers\_usecase.dart'; |
|  | import 'car\_drivers.dart'; |
|  | import 'api/controllers/register\_controller.dart'; |
|  | import 'data/models/user\_model.dart'; |
|  | import 'html\_template.dart'; |
|  |  |
|  | class AppChannel extends ApplicationChannel { |
|  | ManagedContext context; |
|  | AuthServer authServer; |
|  |  |
|  | GetCarUsecase getCarUsecase; |
|  | GetCarsUsecase getCarsUsecase; |
|  | GetDriverUsecase getDriverUsecase; |
|  | GetDriversUsecase getDriversUsecase; |
|  | CreateCarUsecase createCarUsecase; |
|  | CreateDriverUsecase createDriverUsecase; |
|  | DeleteCarUsecase deleteCarUsecase; |
|  | DeleteDriverUsecase deleteDriverUsecase; |
|  | CollectRandomPairUsecase collectRandomPairUsecase; |
|  |  |
|  | final HTMLRenderer htmlRenderer = HTMLRenderer(); |
|  |  |
|  | @override |
|  | SecurityContext get securityContext { |
|  | return SecurityContext() |
|  | //..usePrivateKey("lib/certs/server\_localhost.key", password: "1234") |
|  | //..useCertificateChain("lib/certs/server\_localhost.crt", password: "1234") |
|  | ; |
|  | } |
|  |  |
|  | @override |
|  | Future prepare() async { |
|  | logger.onRecord.listen( |
|  | (rec) => print("$rec ${rec.error ?? ""} ${rec.stackTrace ?? ""}")); |
|  | final dataModel = ManagedDataModel.fromCurrentMirrorSystem(); |
|  | final persistentStore = PostgreSQLPersistentStore.fromConnectionInfo( |
|  | "admin", |
|  | "admin", |
|  | "localhost", |
|  | 5432, |
|  | "drivers", |
|  | ); |
|  |  |
|  | context = ManagedContext(dataModel, persistentStore); |
|  |  |
|  | final authStorage = ManagedAuthDelegate<UserModel>(context); |
|  | authServer = AuthServer(authStorage); |
|  |  |
|  | final CarDatasource carDatasource = CarDatasource( |
|  | context, |
|  | ); |
|  | final DriverDatasource driverDatasource = DriverDatasource( |
|  | context, |
|  | ); |
|  | final IDriverRepository driverRepository = DriverRepository( |
|  | driverDatasource: driverDatasource, |
|  | ); |
|  | final ICarRepository carRepository = CarRepository( |
|  | carDatasource: carDatasource, |
|  | ); |
|  | getCarUsecase = GetCarUsecase( |
|  | carRepository, |
|  | ); |
|  | getCarsUsecase = GetCarsUsecase( |
|  | carRepository, |
|  | ); |
|  | getDriverUsecase = GetDriverUsecase( |
|  | driverRepository, |
|  | ); |
|  | getDriversUsecase = GetDriversUsecase( |
|  | driverRepository, |
|  | ); |
|  | createCarUsecase = CreateCarUsecase( |
|  | carRepository, |
|  | ); |
|  | createDriverUsecase = CreateDriverUsecase( |
|  | driverRepository, |
|  | ); |
|  | deleteCarUsecase = DeleteCarUsecase( |
|  | carRepository, |
|  | ); |
|  | deleteDriverUsecase = DeleteDriverUsecase( |
|  | driverRepository, |
|  | ); |
|  | collectRandomPairUsecase = CollectRandomPairUsecase( |
|  | carRepository, |
|  | driverRepository, |
|  | ); |
|  | } |
|  |  |
|  | @override |
|  | Controller get entryPoint { |
|  | final router = Router(); |
|  |  |
|  | router.route('/auth/token').link( |
|  | () => AuthController( |
|  | authServer, |
|  | ), |
|  | ); |
|  |  |
|  | router |
|  | .route("/drivers/[:id]") |
|  | //.link(() => Authorizer.bearer(authServer)) |
|  | .link( |
|  | () => DriversController( |
|  | getDriverUsecase, |
|  | getDriversUsecase, |
|  | deleteDriverUsecase, |
|  | createDriverUsecase, |
|  | ), |
|  | ); |
|  |  |
|  | router |
|  | .route('/cars/[:id]') |
|  | //.link(() => Authorizer.bearer(authServer)) |
|  | .link( |
|  | () => CarsController( |
|  | createCarUsecase, |
|  | deleteCarUsecase, |
|  | getCarUsecase, |
|  | getCarsUsecase, |
|  | ), |
|  | ); |
|  |  |
|  | router |
|  | .route('/random') |
|  | //.link(() => Authorizer.bearer(authServer)) |
|  | .link( |
|  | () => RandomController( |
|  | collectRandomPairUsecase, |
|  | ), |
|  | ); |
|  |  |
|  | router.route('/register').link( |
|  | () => RegisterController( |
|  | context, |
|  | authServer, |
|  | ), |
|  | ); |
|  |  |
|  | return router; |
|  | } |
|  |  |
|  | @override |
|  | Future<String> render(AuthRedirectController forController, Uri requestUri, |
|  | String responseType, String clientID, String state, String scope) async { |
|  | final map = { |
|  | "response\_type": responseType, |
|  | "client\_id": clientID, |
|  | "state": state |
|  | }; |
|  | map["path"] = requestUri.path; |
|  | if (scope != null) { |
|  | map["scope"] = scope; |
|  | } |
|  |  |
|  | return htmlRenderer.renderHTML("web/login.html", map); |
|  | } |
|  | } |

CONTROLLERS:

|  |  |
| --- | --- |
| import 'package:car\_drivers/data/mappers/car\_mapper.dart'; | |
|  | | import 'package:car\_drivers/domain/models/add\_car\_model.dart'; | |
|  | | import 'package:car\_drivers/domain/usecase/create\_car\_usecase.dart'; | |
|  | | import 'package:car\_drivers/domain/usecase/delete\_car\_usecase.dart'; | |
|  | | import 'package:car\_drivers/domain/usecase/get\_car\_usecase.dart'; | |
|  | | import 'package:car\_drivers/domain/usecase/get\_cars\_usecase.dart'; | |
|  | |  | |
|  | | import '../../car\_drivers.dart'; | |
|  | |  | |
|  | | class CarsController extends ResourceController { | |
|  | | CarsController( | |
|  | | this.createCarUsecase, | |
|  | | this.deleteCarUsecase, | |
|  | | this.getCarUsecase, | |
|  | | this.getCarsUsecase, | |
|  | | ); | |
|  | |  | |
|  | | final GetCarUsecase getCarUsecase; | |
|  | | final GetCarsUsecase getCarsUsecase; | |
|  | | final CreateCarUsecase createCarUsecase; | |
|  | | final DeleteCarUsecase deleteCarUsecase; | |
|  | |  | |
|  | | /// Получить все авто | |
|  | | @Operation.get() | |
|  | | Future<Response> getAllCars() async { | |
|  | |  | |
|  | | final result = await getCarsUsecase(); | |
|  | |  | |
|  | | return Response.ok(CarMapper.mapEntitiesToModels(result), headers: {'Content-type': 'application/json'}); | |
|  | | } | |
|  | |  | |
|  | | /// Получить авто по его id | |
|  | | @Operation.get('id') | |
|  | | Future<Response> getCarByID(@Bind.path('id') int id) async { | |
|  | |  | |
|  | | final result = await getCarUsecase(id); | |
|  | | return Response.ok(CarMapper.mapEntityToModel(result), headers: {'Content-type': 'application/json'}); | |
|  | | } | |
|  | |  | |
|  | | @Operation.post() | |
|  | | Future<Response> addCar( | |
|  | | @Bind.body() AddCarModel input | |
|  | | ) async { | |
|  | |  | |
|  | | final result = await createCarUsecase(input); | |
|  | |  | |
|  | | return Response.ok(CarMapper.mapEntityToModel(result), headers: {'Content-type': 'application/json'}); | |
|  | | } | |
|  | |  | |
|  | | @Operation.delete('id') | |
|  | | Future<Response> deleteCar( | |
|  | | @Bind.path('id') int carID, | |
|  | | ) async { | |
|  | |  | |
|  | | final result = await deleteCarUsecase(carID); | |
|  | |  | |
|  | | return Response.ok(result); | |
|  | | } | |
|  | | } | |
| import 'package:car\_drivers/data/mappers/driver\_mapper.dart'; | | |
|  | | | import 'package:car\_drivers/domain/models/add\_driver\_model.dart'; |
|  | | | import 'package:car\_drivers/domain/usecase/create\_driver\_usecase.dart'; |
|  | | | import 'package:car\_drivers/domain/usecase/delete\_driver\_usecase.dart'; |
|  | | | import 'package:car\_drivers/domain/usecase/get\_driver\_usecase.dart'; |
|  | | | import 'package:car\_drivers/domain/usecase/get\_drivers\_usecase.dart'; |
|  | | |  |
|  | | | import '../../car\_drivers.dart'; |
|  | | |  |
|  | | | class DriversController extends ResourceController { |
|  | | |  |
|  | | | DriversController( |
|  | | | this.getDriverUsecase, |
|  | | | this.getDriversUsecase, |
|  | | | this.deleteDriverUsecase, |
|  | | | this.createDriverUsecase, |
|  | | | ); |
|  | | |  |
|  | | | final GetDriverUsecase getDriverUsecase; |
|  | | | final GetDriversUsecase getDriversUsecase; |
|  | | | final DeleteDriverUsecase deleteDriverUsecase; |
|  | | | final CreateDriverUsecase createDriverUsecase; |
|  | | |  |
|  | | | /// Получить всех водителей |
|  | | | @Operation.get() |
|  | | | Future<Response> getAllDrivers() async { |
|  | | | final result = await getDriversUsecase(); |
|  | | | return Response.ok(DriverMapper.mapEntitiesToModels(result), headers: {'Content-type': 'application/json'}); |
|  | | | } |
|  | | |  |
|  | | | /// Получить водителя по его id |
|  | | | @Operation.get('id') |
|  | | | Future<Response> getDriverByID(@Bind.path('id') int id) async { |
|  | | | final result = await getDriverUsecase(id); |
|  | | | return Response.ok(DriverMapper.mapEntityToModel(result), headers: {'Content-type': 'application/json'}); |
|  | | | } |
|  | | |  |
|  | | | @Operation.post() |
|  | | | Future<Response> addDriver( |
|  | | | @Bind.body() AddDriverModel input, |
|  | | | ) async { |
|  | | |  |
|  | | | final result = await createDriverUsecase(input); |
|  | | |  |
|  | | | return Response.ok(DriverMapper.mapEntityToModel(result), headers: {'Content-type': 'application/json'}); |
|  | | | } |
|  | | |  |
|  | | | @Operation.delete('id') |
|  | | | Future<Response> deleteDriver( |
|  | | | @Bind.path('id') int driverID, |
|  | | | ) async { |
|  | | |  |
|  | | | final result = await deleteDriverUsecase(driverID); |
|  | | |  |
|  | | | return Response.ok(result); |
|  | | | } |
|  | | | } |
| import 'dart:math'; |
|  |  | | |
|  | import 'package:car\_drivers/data/mappers/car\_mapper.dart'; | | |
|  | import 'package:car\_drivers/data/mappers/driver\_mapper.dart'; | | |
|  | import 'package:car\_drivers/domain/usecase/collect\_random\_pair\_usecase.dart'; | | |
|  |  | | |
|  | import '../../car\_drivers.dart'; | | |
|  |  | | |
|  | class RandomController extends ResourceController { | | |
|  |  | | |
|  | RandomController(this.collectRandomPairUsecase); | | |
|  |  | | |
|  | final CollectRandomPairUsecase collectRandomPairUsecase; | | |
|  |  | | |
|  | /// Получить рандомную пару водитель-авто | | |
|  | @Operation.get() | | |
|  | Future<Response> getRandomMatch() async { | | |
|  |  | | |
|  | final result = await collectRandomPairUsecase(); | | |
|  |  | | |
|  | // тут конечно не фонтан, лень было | | |
|  | return Response.ok( | | |
|  | { | | |
|  | "car": CarMapper.mapEntityToModel(result.car)?.asMap(), | | |
|  | "driver": DriverMapper.mapEntityToModel(result.driver)?.asMap(), | | |
|  | }, | | |
|  | headers: {'Content-type': 'application/json'}, | | |
|  | ); | | |
|  | } | | |
|  | } | | |

|  |
| --- |
| import 'dart:async'; |
|  | import 'package:car\_drivers/car\_drivers.dart'; | |
|  | import 'package:car\_drivers/data/models/user\_model.dart'; | |
|  |  | |
|  | class RegisterController extends ResourceController { | |
|  | RegisterController(this.context, this.authServer); | |
|  |  | |
|  | final ManagedContext context; | |
|  | final AuthServer authServer; | |
|  |  | |
|  | @Operation.post() | |
|  | Future<Response> createUser(@Bind.body() UserModel user) async { | |
|  | if (user.username == null || user.password == null) { | |
|  | return Response.badRequest( | |
|  | body: {"error": "username and password required."}); | |
|  | } | |
|  |  | |
|  | user | |
|  | ..salt = AuthUtility.generateRandomSalt() | |
|  | ..hashedPassword = authServer.hashPassword(user.password, user.salt); | |
|  |  | |
|  | return Response.ok(await Query(context, values: user).insert()); | |
|  | } | |
|  | } | |
| import '../../car\_drivers.dart'; | |
|  | |  |
|  | | class UselessPageController extends ResourceController { |
|  | |  |
|  | | @Operation.get() |
|  | | Future<Response> getAllDrivers( |
|  | | {@Bind.header('Accept-Language') String lang} |
|  | | ) async { |
|  | | Map data = { |
|  | | 'Title': 'Zdarova' |
|  | | }; |
|  | | if(lang?.contains('ru') ?? false) { |
|  | | data['title'] = 'Здарова'; |
|  | | } |
|  | | return Response.ok( |
|  | | """ |
|  | | <!DOCTYPE html> |
|  | | <html lang="ru"> |
|  | |  |
|  | | <head> |
|  | | <meta charset="UTF-8"> |
|  | | <title>Авторизация</title> |
|  | | </head> |
|  | |  |
|  | | <body> |
|  | | <div class="container"> |
|  | | <h1>$data</h1> |
|  | | </div> |
|  | | </body> |
|  | |  |
|  | | </html> |
|  | | """, |
|  | | )..contentType = ContentType.html; |
|  | | } |
|  | | } |

**Вывод:**в ходе выполнения данной лабораторной работы мы разработали интерфейс для работы с приложением.