Лабораторная работа №4

TypeScript, Angular

1 часть. На основе предметной области согласно своего варианта спроектировать **Angular** приложение.
Сгенерировать шаблон-заготовку приложения можно при помощи команды angular cli:
ng new <имя_приложения>

Корневой компонент приложения - app.component (генерируется автоматически при создании приложения и размещается в папке src\app). В app.component.html (html шаблоне корневого компонента) разместить bootstrap компонент jumbotron, с описанием функционала приложения (необходимо также добавить bootstrap зависимость в файл package.json).
В папке проекта src\app сгенерировать **модуль** с названием согласно варианта задания. Это можно сделать командой angular cli:
ng generate module <имя_модуля> --module=app

Перейти в папку src\app\имя_модуля и сгенерировать **компоненты** согласно варианта задания:
ng generate component <имя_компонента>

Для взаимодействия с данными в папке src\app\имя_модуля создать папку services и в ней сгенерировать **сервис** при помощи команды:
ng generate service <имя_модуля>

Для осуществления **маршрутизации** в приложении в корень проекта (на одном уровне с app.component) добавить модуль AppRoutingModule. Это можно сделать командой:
ng generate module app-routing --flat --module=app
В app.component.html разместить директиву <router-outlet> </router-outlet> (На место элемента <router-outlet> будет рендериться компонент, выбранный для обработки запроса.).
Для обработки дочерних маршрутов в папке src\app\имя_модуля добавить модуль имя_модуляRoutingModule командой
ng generate module имя_модуля-routing --flat --module= имя_модуля

2 часть. Разработать форму (template-driven form) для добавления новых элементов (согласно варианта), обновления и удаления элементов. Добавить ссылку на форму в приложение.

3 часть. Развернуть приложение на облачной платформе Firebase (см. руководство Firebase). Добавить к Firebase проекту NoSQL базу данных Cloud Firestore (см. руководство Cloud Firestore). В базе Cloud Firestore создать коллекцию согласно варианта задания, например, list-subjects. В коллекцию добавить документы, с полями, соответствующими варианту.

	Срок выдачи задания:	Срок сдачи задания:
12 группа	04.09.23	14.09.23 – структура приложения, навигация; 21.09.23 - форма для редактирования; 28.09.23 – проект на Firebase
13 группа	04.09.23	11.09.23 – структура приложения, навигация; 18.05.23 - форма для добавления элементов; 25.09.23 – проект на Firebase

14 группа	04.09.23	11.09.23 – структура приложения, навигация; 18.05.23 - форма для добавления элементов; 25.09.23 – проект на Firebase

Максимальное количество баллов за работу – 60.

Дата сдачи
До указанной даты – коэффициент 1, до 24.09 – коэффициент 0,6, позже – коэффициент 0,4.

Варианты:

1. **Факультатив**. Создать модуль (module) с названием Subjects. В этом модуле разместить компоненты (component) SubjectCenter, SubjectList, SubjectDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент SubjectCenter, для которого дочерним подгружается компонент SubjectList. При нажатии на один из элементов списка SubjectList подгружается соответствующий компонент SubjectDetails.
   Список курсов хранить в приложении в файле mock-subject-list.ts в виде массива: Subjects: Subject [] = [{ id: 1, name: 'Math', teacher: 'Drozd' }, …]

2. **Платежи.** Создать модуль (module) с названием Accounts. В этом модуле разместить компоненты (component) AccountCenter, AccountList, AccountDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент AccountCenter, для которого дочерним подгружается компонент AccountList. При нажатии на один из элементов списка AccountList подгружается соответствующий компонент AccountDetails. Список счетов хранить в приложении в файле mock-account-list.ts. в виде массива:
   Accounts: Account [] = [{ id: 1, sum: 'Math', owner: 'Drozd' }, …]

3. **Больница**. Создать модуль (module) с названием Prescriptions. В этом модуле разместить компоненты (component) PrescriptionCenter, PrescriptionList, PrescriptionDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент PrescriptionCenter, для которого дочерним подгружается компонент PrescriptionList. При нажатии на один из элементов списка PrescriptionList подгружается соответствующий компонент PrescriptionDetails.
   Список назначений хранить в приложении в файле mock-prescription-list.ts в виде массива: Prescriptions: Prescription [] = [{ id: 1, description: 'pill', doctor: 'Drozd', patient: ‘Linev’ }, …]

4. **Вступительные экзамены**. Создать модуль (module) с названием Exams. В этом модуле разместить компоненты (component) ExamCenter, ExamList, ExamDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент ExamCenter, для которого дочерним подгружается компонент ExamList. При нажатии на один из элементов списка ExamList подгружается соответствующий компонент ExamDetails.
   Список результатов экзаменов хранить в приложении в файле mock-exam-list.ts в виде массива:
   Exams: Exam [] = [{ id: 1, name: 'Math', mark: '10', enrollee: ‘Rudnev’ }, …]

5. **Библиотека**. Создать модуль (module) с названием Books. В этом модуле разместить компоненты (component) BookCenter, BookList, BookDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент BookCenter,

для которого дочерним подгружается компонент BookList. При нажатии на один из элементов списка BookList подгружается соответствующий компонент BookDetails.
Список книг хранить в приложении в файле mock-book-list.ts в виде массива: Books: Book [] = [{ id: 1, name: 'Angular', author: 'Drozd' }, …]

6. **Конструкторское бюро**. Создать модуль (module) с названием Projects. В этом модуле разместить компоненты (component) ProjectCenter, ProjectList, ProjectDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент ProjectCenter, для которого дочерним подгружается компонент ProjectList. При нажатии на один из элементов списка ProjectList подгружается соответствующий компонент ProjectDetails.
Список проектов хранить в приложении в файле mock-project-list.ts в виде массива: Projects: Project [] = [{ id: 1, description: 'progr', customer: 'Drozd', price: 123 }, …]

7. **Телефонная станция**. Создать модуль (module) с названием Services. В этом модуле разместить компоненты (component) ServiceCenter, ServiceList, ServiceDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент ServiceCenter, для которого дочерним подгружается компонент ServiceList. При нажатии на один из элементов списка ServiceList подгружается соответствующий компонент ServicetDetails.
Список сервисов хранить в приложении в файле mock-service-list.ts в виде массива: Services: Service [] = [{ id: 1, description: 'Internet', price: 123 }, …]

8. **Автобаза**. Создать модуль (module) с названием Trips. В этом модуле разместить компоненты (component) TripCenter, TripList, TripDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент TripCenter, для которого дочерним подгружается компонент TripList. При нажатии на один из элементов списка TripList подгружается соответствующий компонент TripDetails.
Список рейсов хранить в приложении в файле mock-trip-list.ts в виде массива: Trips: Trip [] = [{ id: 1, description: 'freight', driver: 'Drozd' }, …]

9. **Интернет-магазин**. Создать модуль (module) с названием Goods. В этом модуле разместить компоненты (component) GoodCenter, GoodList, GoodDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент GoodCenter, для которого дочерним подгружается компонент GoodList. При нажатии на один из элементов списка GoodList подгружается соответствующий компонент GoodDetails.
Список товаров хранить в приложении в файле mock-good-list.ts в виде массива: Goods: Good [] = [{ id: 1, description: 'pets', price: '123' }, …]

10. Система **Ресторан**. Создать модуль (module) с названием Menu. В этом модуле разместить компоненты (component) MenuCenter, MenuList, MenuDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент MenuCenter, для которого дочерним подгружается компонент MenuList. При нажатии на один из элементов списка MenuList подгружается соответствующий компонент MenuDetails.
Меню хранить в приложении в файле mock-menu-list.ts в виде массива: Menus: Menu [] = [{ id: 1, description: 'pie', price: 'Drozd' }, …]

11. **Авиакомпания.** Создать модуль (module) с названием Flights. В этом модуле разместить компоненты (component) FlightCenter, FlightList, FlightDetails. Добавить

навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент FlightCenter, для которого дочерним подгружается компонент FlightList. При нажатии на один из элементов списка FlightList подгружается соответствующий компонент FlightDetails.
Список рейсов хранить в приложении в файле mock-flight-list.ts в виде массива: Flights: Flight [] = [{ id: 1, from: 'Minsk', to: ‘London’, crew: 'asd' }, …]

12. **Периодические издания**. Создать модуль (module) с названием Periodicals. В этом модуле разместить компоненты (component) PeriodicalCenter, PeriodicalList, PeriodicalDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент PeriodicalCenter, для которого дочерним подгружается компонент PeriodicalList. При нажатии на один из элементов списка PeriodicalList подгружается соответствующий компонент PeriodicalDetails.
Список изданий хранить в приложении в файле mock-periodical-list.ts в виде массива: Periodicals: Periodical [] = [{ id: 1, name: 'News', publisher: 'Drozd', price: 123 }, …]

13. **Заказ гостиницы**. Создать модуль (module) с названием Apartments. В этом модуле разместить компоненты (component) ApartmentCenter, ApartmentList, ApartmentDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент ApartmentCenter, для которого дочерним подгружается компонент ApartmentList. При нажатии на один из элементов списка ApartmentList подгружается соответствующий компонент ApartmentDetails.
Список номеров хранить в приложении в файле mock-apartment-list.ts в виде массива: Apartments: Apartment [] = [{ id: 1, description: 'vip', price: 123 }, …]

14. **Жилищно-коммунальные услуги**. Создать модуль (module) с названием Housings. В этом модуле разместить компоненты (component) HousingCenter, HousingList, HousingDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент HousingCenter, для которого дочерним подгружается компонент HousingList. При нажатии на один из элементов списка HousingList подгружается соответствующий компонент HousingDetails.
Список работ хранить в приложении в файле mock-housing-list.ts в виде массива: Housings: Housing [] = [{ id: 1, description: 'electricity', worker: 'Drozd' }, …]

15. **Прокат автомобилей**. Создать модуль (module) с названием Cars. В этом модуле разместить компоненты (component) CarCenter, CarList, CarDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент CarCenter, для которого дочерним подгружается компонент CarList. При нажатии на один из элементов списка CarList подгружается соответствующий компонент CarDetails.
Список автомобилей хранить в приложении в файле mock-car-list.ts в виде массива: Cars: Car [] = [{ id: 1, model: 'BMW', state: 'good', price: 123 }, …]

16. **Скачки.** Создать модуль (module) с названием Races. В этом модуле разместить компоненты (component) RaceCenter, RaceList, RaceDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент RaceCenter, для которого дочерним подгружается компонент RaceList. При нажатии на один из элементов списка RaceList подгружается соответствующий компонент RaceDetails.
Список забегов хранить в приложении в файле mock-race-list.ts в виде массива: Races: Race [] = [{ id: 1, date: 1.03, winner: 'Luch' }, …]
может делать **Ставки** на разных **Лошадей Забега**. **Лошадь** не может

17. **Тестирование**. Создать модуль (module) с названием Tests. В этом модуле разместить компоненты (component) TestCenter, TestList, TestDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент TestCenter, для которого дочерним подгружается компонент TestList. При нажатии на один из элементов списка TestList подгружается соответствующий компонент TestDetails.
Список тестов хранить в приложении в файле mock-test-list.ts в виде массива: Tests: Test [] = [{ id: 1, subject: 'Math', complexity: 'medium' }, …]

18. **Кофе-машина**. Создать модуль (module) с названием Drinks. В этом модуле разместить компоненты (component) DrinkCenter, DrinkList, DrinkDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент DrinkCenter, для которого дочерним подгружается компонент DrinkList. При нажатии на один из элементов списка DrinkList подгружается соответствующий компонент DrinkDetails.
Список напитков хранить в приложении в файле mock-drink-list.ts в виде массива: Drinks: Drink [] = [{ id: 1, name: 'Latte', price: 123 }, …]

19. **Парк**. Создать модуль (module) с названием Plants. В этом модуле разместить компоненты (component) PlantCenter, PlantList, PlantDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент PlantCenter, для которого дочерним подгружается компонент PlantList. При нажатии на один из элементов списка PlantList подгружается соответствующий компонент PlantDetails.
Список растений хранить в приложении в файле mock-plant-list.ts в виде массива: Plants: Plant [] = [{ id: 1, name: 'tree', task: 'watering' }, …]

20. **Турагентство**. Создать модуль (module) с названием Tours. В этом модуле разместить компоненты (component) TourCenter, TourList, TourDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент TourCenter, для которого дочерним подгружается компонент TourList. При нажатии на один из элементов списка TourList подгружается соответствующий компонент TourDetails.
Список туров хранить в приложении в файле mock-tour-list.ts в виде массива: Tours: Tour [] = [{ id: 1, destination: 'Paris', type: 'excursion' }, …]

21. Система **Команда разработчиков**. Создать модуль (module) с названием Developers. В этом модуле разместить компоненты (component) DeveloperCenter, DeveloperList, DeveloperDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент DeveloperCenter, для которого дочерним подгружается компонент DeveloperList. При нажатии на один из элементов списка DeveloperList подгружается соответствующий компонент DeveloperDetails.
Список разработчиков хранить в приложении в файле mock-developer-list.ts в виде массива: Developers: Developer [] = [{ id: 1, name: 'Drozd', qualification: 'manager', salary: 123 }, …]

22. **Видеотека**. Создать модуль (module) с названием Movies. В этом модуле разместить компоненты (component) MovieCenter, MovieList, MovieDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент MovieCenter, для которого дочерним подгружается компонент MovieList. При нажатии на один из элементов списка MovieList подгружается соответствующий компонент MovieDetails.
Список фильмов хранить в приложении в файле mock-movie-list.ts в виде массива: Movies: Movie [] = [{ id: 1, name: 'Math', direcor: 'Drozd'. date: 1.02 }, …]

23. **Расписание занятий.** Создать модуль (module) с названием Subjects. В этом модуле разместить компоненты (component) SubjectCenter, SubjectList, SubjectDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент SubjectCenter, для которого дочерним подгружается компонент SubjectList. При нажатии на один из элементов списка SubjectList подгружается соответствующий компонент SubjectDetails.

Список дисциплин хранить в приложении в файле mock-subject-list.ts в виде массива: Subjects: Subject [] = [{ id: 1, name: 'Math', teacher: 'Drozd', dayWeek: ‘monday’, class: 123 }, …]

24. **Погода.** Создать модуль (module) с названием Regions. В этом модуле разместить компоненты (component) RegionCenter, RegionList, RegionDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент RegionCenter, для которого дочерним подгружается компонент RegionList. При нажатии на один из элементов списка RegionList подгружается соответствующий компонент RegionDetails.

Список регионов хранить в приложении в файле mock-region-list.ts в виде массива: Regions: Region [] = [{ id: 1, name: 'Gomel', language: ‘russian’, square: 123 }, …]

25. Система **LowCost-Авиакомпания**. Создать модуль (module) с названием Flights. В этом модуле разместить компоненты (component) FlightCenter, FlightList, FlightDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент FlightCenter, для которого дочерним подгружается компонент FlightList. При нажатии на один из элементов списка FlightList подгружается соответствующий компонент FlightDetails.

Список рейсов хранить в приложении в файле mock-flight-list.ts в виде массива: Flights: Flight [] = [{ id: 1, from: 'Minsk', to: ‘London’, date: 1.03, luggage: 20, price: 80 }, …]

26. Зоопарк. Создать модуль (module) с названием Zoo. В этом модуле разместить компоненты (component) AnimalCenter, AnimalList, AnimalDetails. Добавить навигацию: при открытии проекта на место элемента <router-outlet> отображается компонент AnimalCenter, для которого дочерним подгружается компонент AnimalList. При нажатии на один из элементов списка AnimalList подгружается соответствующий компонент AnimalDetails.

Список животных хранить в приложении в файле mock-animal-list.ts в виде массива:
Animals: Animal [] = [{ id: 1, animalkind:’Panda’, nikname: 'Boo', age: ‘3’, weight: ‘12’ }, …]