МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»  
(Самарский университет)

Институт информатики и кибернетики

Кафедра программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
по дисциплине «Frontend-разработка»

по теме «Работа с клиентами»

Выполнил:

Обучающийся группы № 6415-020302D И.А. Яшин

Проверил: М.О. Бочкарев

Дата защиты

Оценка

Самара 2024

Задание

Реализовать сайт для работы с клиентами, при этом должны быть реализованы методы CRUD следующих сущностей:

* Клиент (Customer): Сущность, представляющая клиента компании. Включает в себя данные, такие как имя, фамилия, контактная информация, история покупок и статус клиента;
* Сделка (Deal): Сущность, представляющая сделку или сделку с клиентом. Включает в себя данные о товарах или услугах, цене, сроках и статусе сделки;
* Жалоба (Complaint): Сущность, представляющая жалобу или проблему клиента. Включает в себя данные, такие как описание проблемы, статус жалобы и дата открытия.

Описание предметной области

Клиент — это главный участник платформы, представляющий собой физическое или юридическое лицо, с которым компания имеет дело. Именно через эту сущность происходит взаимодействие с продуктами или услугами. Информация о клиенте включает его личные данные, такие как имя, фамилия, адрес электронной почты и номер телефона, а также данные о его предыдущих покупках. Важным аспектом является также статус клиента, который может определять его значимость для компании, а также его потенциальную степень лояльности. Управление клиентами включает в себя не только предоставление им товаров или услуг, но и поддержку, учет и анализ их активности для оптимизации взаимоотношений и увеличения уровня удовлетворенности.

Сделка представляет собой коммерческое соглашение между компанией и клиентом, в результате которого происходит продажа товара или услуги. Эта сущность описывает все детали сделки, включая перечень товаров или услуг, их стоимость, сроки выполнения и текущий статус. Сделка может иметь различные этапы от начала до завершения, и управление этим процессом помогает не только контролировать текущие продажи, но и анализировать и улучшать стратегии продаж в будущем. Отслеживание сделок также позволяет компании прогнозировать свои доходы и планировать деятельность на основе текущих и будущих продаж.

Жалоба представляет собой отзыв или проблему, высказанную клиентом в отношении продукта, услуги или обслуживания компании. Она может быть связана с качеством товара, сроками исполнения услуги, обслуживанием клиента и т. д. Управление жалобами включает в себя прием, регистрацию, анализ и разрешение проблемы, обеспечивая клиенту удовлетворительное решение и предотвращая возможные негативные последствия для репутации компании. Эффективное управление жалобами также позволяет компании выявлять проблемные области в своей деятельности и принимать меры по их устранению, что способствует улучшению общего качества продукции или услуги.

Описание процесса реализации

При разработке использовались следующие технологии и иснтрументы:

* фронтенд: разработка выполнена с использованием Angular, популярного фреймворка для создания веб-приложений на TypeScript;
* языки программирования: TypeScript и HTML для структуры и логики приложения, CSS для стилизации;
* базовый сервер: для эмуляции базы данных используется json-server, который позволяет работать с JSON файлами как с базой данных;
* управление версиями: Git используется для контроля версий кода, что обеспечивает возможность отслеживания изменений и совместной работы над проектом.

Были созданы следующие компоненты:

* главный компонент (AppComponent): служит точкой входа в приложение, отображает основное меню и содержимое страниц;
* компоненты для управления данными (CustomerComponent, DealComponent, ComplaintComponent): каждый из этих компонентов отвечает за отображение и управление соответствующими данными (клиенты, сделки, жалобы). Они включают в себя формы для добавления, редактирования и удаления записей;
* модели данных (Customer, Deal, Complaint): определяют структуру данных для клиентов, сделок и жалоб соответственно;
* роутинг (AppRoutingModule): определяет маршруты приложения, позволяя пользователю переходить между различными компонентами.

В процессе разработки были пройдены следующие этапы:

* настройка проекта: использование Angular CLI для создания нового проекта и настройки его структуры;
* разработка компонентов: создание компонентов для отображения и управления данными, включая формы для ввода данных;
* работа с данными: использование json-server для эмуляции базы данных и взаимодействия с ней через HTTP запросы;
* тестирование: регулярное тестирование компонентов и функциональности приложения для обеспечения корректной работы;
* управление версиями: использование Git для отслеживания изменений в коде и управления версиями.

Экранная форма одной из страниц представлена на рисунке 1.

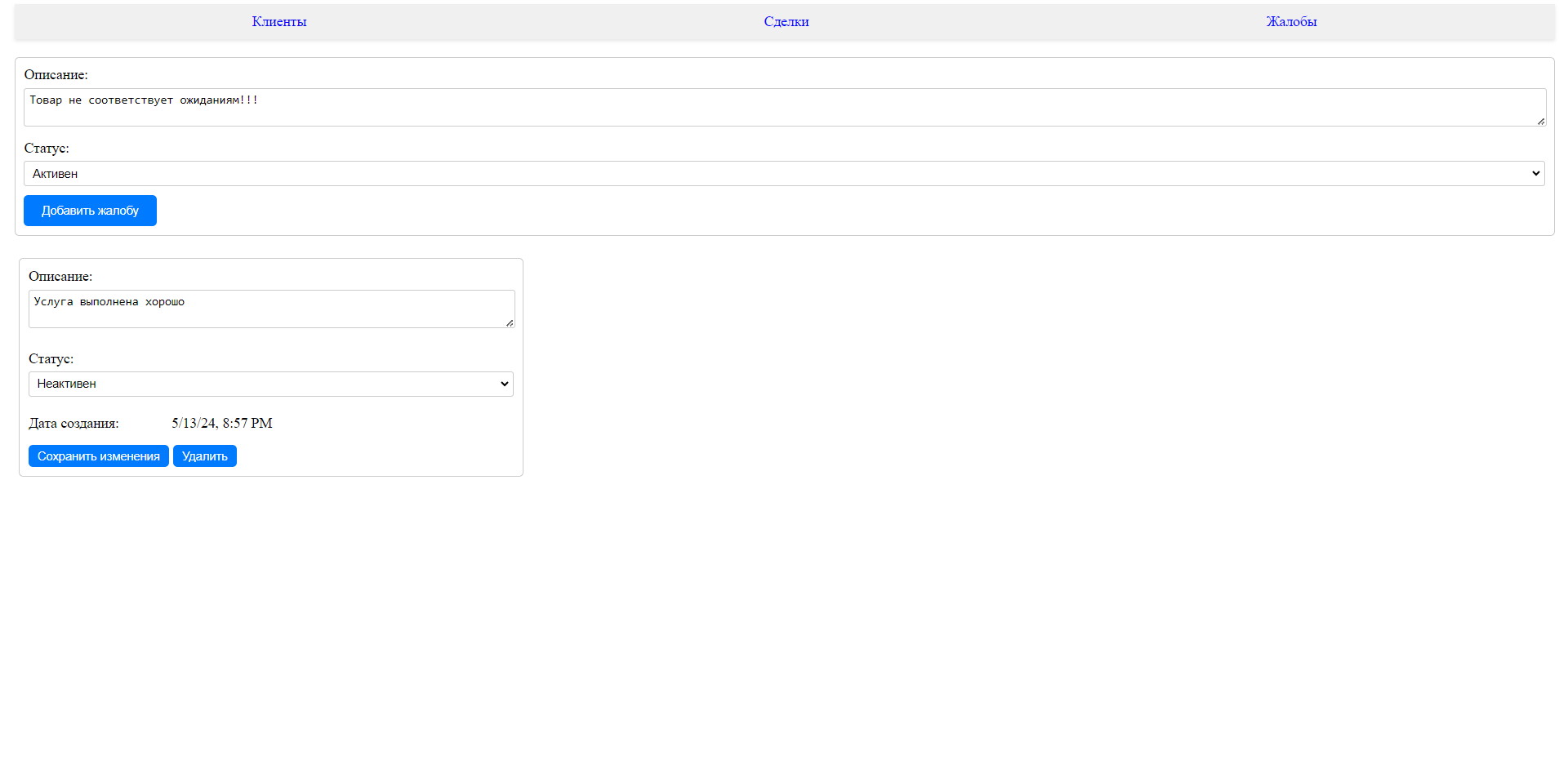


Рисунок 1 – Страница управления жалобами

Вывод

В процессе выполнения задания были успешно освоены и развиты следующие технологии и навыки.

Технологии:

* Angular: изучены основы и продвинутые аспекты Angular, включая создание компонентов, модулей, сервисов и директив. Это позволяет понять, как строить масштабируемые и модульные веб-приложения;
* TypeScript: использование TypeScript для разработки приложения даёт глубокое понимание типизации в JavaScript, что является ключевым навыком для современной разработки;
* HTTP и REST API: приобретены навыки работаты с HTTP запросами и REST API для взаимодействия с сервером, что является важным аспектом разработки веб-приложений;
* JSON и JSON Server: использование `json-server` для эмуляции базы данных позволяет понять, как работать с JSON данными и как они могут быть использованы в веб-приложениях;
* Git: освоены основы работы с Git для контроля версий, что является неотъемлемой частью современной разработки программного обеспечения.

Навыки:

* разработка компонентов: получены навыки разработки компонентов Angular, которые могут быть повторно использованы в различных частях приложения, что способствует повышению эффективности разработки;
* работа с данными: получены, обработаны и отображены данные в приложении, что является ключевым навыком для любого веб-разработчика;
* анализ и решение проблем: в процессе разработки возникшие проблемы были успешно решены, что способствует развитию аналитических навыков и способности к решению проблем.

ПРИЛОЖЕНИЕ А  
Листинг программы

<!-- complaint.component.html -->

<div>

    <form (submit)="addComplaint($event)" ngNativeValidate>

        <!-- ID (generated automatically) -->

        <input type="hidden" [(ngModel)]="newComplaint.id" name="id">

        <!-- Description -->

        <label for="description">Описание:</label>

        <textarea id="description" [(ngModel)]="newComplaint.description" name="description" required></textarea>

        <!-- Status -->

        <label for="status">Статус:</label>

        <select id="status" [(ngModel)]="newComplaint.status" name="status" required>

            <option value="active">Активен</option>

            <option value="inactive">Неактивен</option>

        </select>

        <button type="submit">Добавить жалобу</button>

    </form>

    <ul class="list">

        <li \*ngFor="let complaint of complaints">

            <div>

                <form (submit)="updateComplaint($event, complaint)" ngNativeValidate>

                    <!-- Форма редактирования для каждой жалобы -->

                    <div>

                        <label>Описание:</label>

                        <textarea [(ngModel)]="complaint.description" name="description" required></textarea>

                    </div>

                    <div>

                        <label>Статус:</label>

                        <select [(ngModel)]="complaint.status" name="status" required>

                            <option value="active">Активен</option>

                            <option value="inactive">Неактивен</option>

                        </select>

                    </div>

                    <div>

                        <label>Дата создания:</label>

                        <span>{{ complaint.createdAt | date : 'short' }}</span>

                    </div>

                    <button type="submit">Сохранить изменения</button>

                    <button (click)="deleteComplaint(complaint.id)">Удалить</button>

                </form>

            </div>

        </li>

    </ul>

</div>

// complaint.component.ts

import { Component, OnInit } from '@angular/core';

import { Complaint } from './complaint.model';

import { HttpClient } from '@angular/common/http';

@Component({

    selector: 'app-complaint',

    templateUrl: './complaint.component.html'

})

export class ComplaintComponent implements OnInit {

    complaints: Complaint[] = [];

    newComplaint: Complaint = {

        description: '',

        status: '',

        createdAt: new Date()

    } as any;

    constructor(private http: HttpClient) { }

    ngOnInit(): void {

        this.http.get<Complaint[]>('http://localhost:3000/complaints').subscribe(complaints => {

            this.complaints = complaints;

        });

    }

    addComplaint(event: Event): void {

        event.preventDefault();

        this.newComplaint.createdAt = new Date();

        this.http.post('http://localhost:3000/complaints', this.newComplaint).subscribe(

            (response) => {

                console.log('Complaint added successfully', response);

                this.complaints.push(response as any);

            },

            (error) => {

                console.error('Error adding complaint', error);

            }

        )

        this.resetNewComplaint();

    }

    resetNewComplaint(): void {

        this.newComplaint = {

            description: '',

            status: '',

            createdAt: new Date()

        } as any;

    }

    deleteComplaint(id: number): void {

        this.http.delete(`http://localhost:3000/complaints/${id}`).subscribe(

            (response) => {

                console.log('Complaint deleted successfully', response);

                this.complaints = this.complaints.filter(complaint => complaint.id !== id);

            },

            (error) => {

                console.error('Error deleting complaint', error);

            }

        );

    }

    updateComplaint(event: Event, complaint: Complaint): void {

        event.preventDefault();

        this.http.put(`http://localhost:3000/complaints/${complaint.id}`, complaint).subscribe(

            (response) => {

                console.log('Complaint updated successfully', response);

                const index = this.complaints.findIndex(c => c.id === complaint.id);

                if (index !== -1) {

                    this.complaints[index] = response as any;

                }

            },

            (error) => {

                console.error('Error updating complaint', error);

            }

        );

    }

}

<!-- customer.component.html -->

<div>

    <form (submit)="addCustomer($event)" ngNativeValidate>

        <!-- ID (generated automatically) -->

        <input type="hidden" [(ngModel)]="newCustomer.id" name="id">

        <!-- First Name -->

        <label for="firstName">Имя:</label>

        <input type="text" id="firstName" [(ngModel)]="newCustomer.firstName" name="firstName" required>

        <!-- Last Name -->

        <label for="lastName">Фамилия:</label>

        <input type="text" id="lastName" [(ngModel)]="newCustomer.lastName" name="lastName" required>

        <!-- Contact Info -->

        <label for="contactInfo">Контактная информация:</label>

        <input type="text" id="contactInfo" [(ngModel)]="newCustomer.contactInfo" name="contactInfo" required>

        <!-- History -->

        <label for="history">История:</label>

        <textarea id="history" [(ngModel)]="newCustomer.history" name="history"></textarea>

        <!-- Status -->

        <label for="status">Статус:</label>

        <select id="status" [(ngModel)]="newCustomer.status" name="status" required>

            <option value="active">Активен</option>

            <option value="inactive">Неактивен</option>

        </select>

        <button type="submit">Добавить клиента</button>

    </form>

    <ul class="list">

        <li \*ngFor="let customer of customers">

            <div>

                <form (submit)="updateCustomer($event, customer)" ngNativeValidate>

                    <!-- Форма редактирования для каждого клиента -->

                    <div>

                        <label>Имя:</label>

                        <input [(ngModel)]="customer.firstName" name="firstName" required>

                    </div>

                    <div>

                        <label>Фамилия:</label>

                        <input [(ngModel)]="customer.lastName" name="lastName" required>

                    </div>

                    <div>

                        <label>Контактная информация:</label>

                        <input [(ngModel)]="customer.contactInfo" name="contactInfo" required>

                    </div>

                    <div>

                        <label>История:</label>

                        <textarea [(ngModel)]="customer.history" name="history"></textarea>

                    </div>

                    <div>

                        <label>Статус:</label>

                        <select [(ngModel)]="customer.status" name="status" required>

                            <option value="active">Активен</option>

                            <option value="inactive">Неактивен</option>

                        </select>

                    </div>

                    <button type="submit">Сохранить изменения</button>

                    <button (click)="deleteCustomer(customer.id)">Удалить</button>

                </form>

            </div>

        </li>

    </ul>

</div>

// customer.component.ts

import { Component, OnInit } from '@angular/core';

import { Customer } from './customer.model'; // Убедитесь, что путь к файлу customer.model.ts верный

import { HttpClient } from '@angular/common/http';

@Component({

    selector: 'app-customer',

    templateUrl: './customer.component.html'

})

export class CustomerComponent implements OnInit {

    customers: Customer[] = [];

    newCustomer: Customer = {

        firstName: '',

        lastName: '',

        contactInfo: '',

        history: '',

        status: ''

    } as any;

    constructor(private http: HttpClient) { }

    ngOnInit(): void {

        this.http.get<Customer[]>('http://localhost:3000/customers').subscribe(customers => {

            this.customers = customers;

        });

    }

    addCustomer(event: Event): void {

        event.preventDefault();

        this.http.post('http://localhost:3000/customers', this.newCustomer).subscribe(

            (response) => {

                console.log('Customer added successfully', response);

                this.customers.push(response as any);

            },

            (error) => {

                console.error('Error adding customer', error);

            }

        )

        this.resetNewCustomer();

    }

    resetNewCustomer(): void {

        this.newCustomer = {

            firstName: '',

            lastName: '',

            contactInfo: '',

            history: '',

            status: ''

        } as any;

    }

    deleteCustomer(id: number): void {

        this.http.delete(`http://localhost:3000/customers/${id}`).subscribe(

            (response) => {

                console.log('Customer deleted successfully', response);

                this.customers = this.customers.filter(customer => customer.id !== id);

            },

            (error) => {

                console.error('Error deleting customer', error);

            }

        );

    }

    updateCustomer(event: Event, customer: Customer): void {

        event.preventDefault();

        this.http.put(`http://localhost:3000/customers/${customer.id}`, customer).subscribe(

            (response) => {

                console.log('Customer updated successfully', response);

                const index = this.customers.findIndex(c => c.id === customer.id);

                if (index !== -1) {

                    this.customers[index] = response as any;

                }

            },

            (error) => {

                console.error('Error updating customer', error);

            }

        );

    }

}

<!-- deal.component.html -->

<div>

    <form (submit)="addDeal($event)" ngNativeValidate>

        <!-- ID (generated automatically) -->

        <input type="hidden" [(ngModel)]="newDeal.id" name="id">

        <!-- Description -->

        <label for="description">Описание:</label>

        <textarea id="description" [(ngModel)]="newDeal.description" name="description" required></textarea>

        <!-- Price -->

        <label for="price">Цена:</label>

        <input type="number" id="price" [(ngModel)]="newDeal.price" name="price" required>

        <!-- Finish Date -->

        <label for="finishDate">Дата завершения:</label>

        <input type="date" id="finishDate" [(ngModel)]="newDeal.finishDate" name="finishDate" required>

        <!-- Status -->

        <label for="status">Статус:</label>

        <select id="status" [(ngModel)]="newDeal.status" name="status" required>

            <option value="active">Активен</option>

            <option value="inactive">Неактивен</option>

        </select>

        <button type="submit">Добавить сделку</button>

    </form>

    <ul class="list">

        <li \*ngFor="let deal of deals">

            <div>

                <form (submit)="updateDeal($event, deal)" ngNativeValidate>

                    <!-- Форма редактирования для каждого дела -->

                    <div>

                        <label>Описание:</label>

                        <textarea [(ngModel)]="deal.description" name="description" required></textarea>

                    </div>

                    <div>

                        <label>Цена:</label>

                        <input type="number" [(ngModel)]="deal.price" name="price" required>

                    </div>

                    <div>

                        <label>Дата завершения:</label>

                        <input type="date" [(ngModel)]="deal.finishDate" name="finishDate" required>

                    </div>

                    <div>

                        <label>Статус:</label>

                        <select [(ngModel)]="deal.status" name="status" required>

                            <option value="active">Активен</option>

                            <option value="inactive">Неактивен</option>

                        </select>

                    </div>

                    <button type="submit">Сохранить изменения</button>

                    <button (click)="deleteDeal(deal.id)">Удалить</button>

                </form>

            </div>

        </li>

    </ul>

</div>

// deal.component.ts

import { Component, OnInit } from '@angular/core';

import { Deal } from './deal.model'; // Убедитесь, что путь к файлу deal.model.ts верный

import { HttpClient } from '@angular/common/http';

@Component({

    selector: 'app-deal',

    templateUrl: './deal.component.html'

})

export class DealComponent implements OnInit {

    deals: Deal[] = [];

    newDeal: Deal = {

        description: '',

        price: 0,

        finishDate: new Date(), // Назначаем текущую дату автоматически

        status: ''

    } as any;

    constructor(private http: HttpClient) { }

    ngOnInit(): void {

        this.http.get<Deal[]>('http://localhost:3000/deals').subscribe(deals => {

            this.deals = deals;

        });

    }

    addDeal(event: Event): void {

        event.preventDefault();

        this.http.post('http://localhost:3000/deals', this.newDeal).subscribe(

            (response) => {

                console.log('Deal added successfully', response);

                this.deals.push(response as any);

            },

            (error) => {

                console.error('Error adding deal', error);

            }

        )

        this.resetNewDeal();

    }

    resetNewDeal(): void {

        this.newDeal = {

            description: '',

            price: 0,

            finishDate: new Date(), // Назначаем текущую дату автоматически

            status: ''

        } as any;

    }

    deleteDeal(id: number): void {

        this.http.delete(`http://localhost:3000/deals/${id}`).subscribe(

            (response) => {

                console.log('Deal deleted successfully', response);

                this.deals = this.deals.filter(deal => deal.id !== id);

            },

            (error) => {

                console.error('Error deleting deal', error);

            }

        );

    }

    updateDeal(event: Event, deal: Deal): void {

        event.preventDefault();

        this.http.put(`http://localhost:3000/deals/${deal.id}`, deal).subscribe(

            (response) => {

                console.log('Deal updated successfully', response);

                const index = this.deals.findIndex(c => c.id === deal.id);

                if (index !== -1) {

                    this.deals[index] = response as any;

                }

            },

            (error) => {

                console.error('Error updating deal', error);

            }

        );

    }

}

// app-routing.module.ts

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { CustomerComponent } from './customer/customer.component';

import { ComplaintComponent } from './complaint/complaint.component';

import { DealComponent } from './deal/deal.component';

const routes: Routes = [

  { path: 'customers', component: CustomerComponent },

  { path: 'deals', component: DealComponent },

  { path: 'complaints', component: ComplaintComponent },

  { path: '', redirectTo: '/customers', pathMatch: 'full' }

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

// app.module.ts

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';

import { AppRoutingModule } from './app-routing.module';

import { CustomerComponent } from './customer/customer.component';

import { ComplaintComponent } from './complaint/complaint.component';

import { DealComponent } from './deal/deal.component';

@NgModule({

  declarations: [

    AppComponent,

    CustomerComponent,

    ComplaintComponent,

    DealComponent

  ],

  imports: [

    BrowserModule,

    FormsModule,

    HttpClientModule,

    AppRoutingModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }