# Uade.me - ваш гид в поиске призвания

web-project

Done by Olzhas, Aruzhan, Akzhan

Мы поможем выявить ваши таланты и уникальные способности, которые позволят вам достичь успеха в выбранной сфере деятельности.

Начать

# FrontEnd

onClick events to hit API: Universities, Specialties; ngFor, ngIf

CSS styling

app > uni-list > uni-list.component.html > div > section.filter_bar > div.dropdown > button.dropdown-btn

```
<div>
  <section class="filter_bar">
    <form class="search_tab">
      <img class="search_icon" src="/assets/images/search_icon.png" (click)="filterResults(filter.value)">
    </form>
    <div class="dropdown">
      <button class="dropdown-btn">Специальность</button>
      <div class="dropdown-content">
        <button (click)="returnAll()">All</button>
        <button *ngFor="let specialty of specialties" (click)="filterListBySpecialty(specialty)">{{ specialty.name }}</button>
      </div>
    </div>
    <!-- Add similar dropdowns for "Университет" and "Город" -->
    <div class="dropdown">
      <button class="dropdown-btn">Университет</button>
      <div class="dropdown-content">
        <button (click)="returnAll()">All</button>
        <button *ngFor="let uni of constList" (click)="filterListByUniversity(uni.name)">{{ uni.name }}</button>
      </div>
    </div>
    <div class="dropdown">
      <button class="dropdown-btn">Город</button>
      <div class="dropdown-content">
        <button (click)="returnAll()">All</button>
        <button *ngFor="let city of cities" (click)="filt
      </div>
    </div>
    <div class="dropdown">
      <button class="dropdown-btn">Сортировать</button>
      <div class="dropdown-content">
        <button *ngFor="let item of sortByList" (click)="
      </div>
```

app > uni-list > TS uni-list.component.ts > ...

```
export class UniListComponent  implements OnInit{

  ngOnInit(): void {
    this.getUniversities();
    this.getCities();
    this.getSpecialties();
  }

  private getUniversities(): void {
   if(this.constList.length){
     this.unis = this.constList
     return
   }
   this.uniService.getUniversities().subscribe(unis =>{
     this.unis = unis
     this.constList = this.unis
   })

  }

  private getCities(): void {
    this.cityService.getCities().subscribe(cities => {
      this.cities = cities;
    });
  }

  private getSpecialties(): void {
    this.specService.getSpecialties().subscribe(specs => {
      this.specialties = specs;
    });
  }
```

```
<div class="accordion">
  <button (click)="toggle()" class="accordion-btn">@ Узнат
  <div *ngIf="isActive" class="accordion-content">
    <p>{{consult.phone}}</p>
    <div class="contacts" *ngIf="consult.mail != ''">
      <img src="assets/images/mail.png">
      <p>{{consult.mail}}</p>
    </div>
    <div class="contacts" *ngIf="consult.insta != ''">
      <img src="assets/images/instagram.png">
      <p>{{consult.insta}}</p>
    </div>
    <div class="contacts" *ngIf="consult.telegram != ''">
      <img src="assets/images/telegram.png">
      <p>{{consult.telegram}}</p>
    </div>
    <div class="contacts" *ngIf="consult.whatsApp != ''">
      <img src="assets/images/whatsapp.png">
      <a href="https://wa.me/{{consult.whatsApp}}?text=Здр
    </div>
```

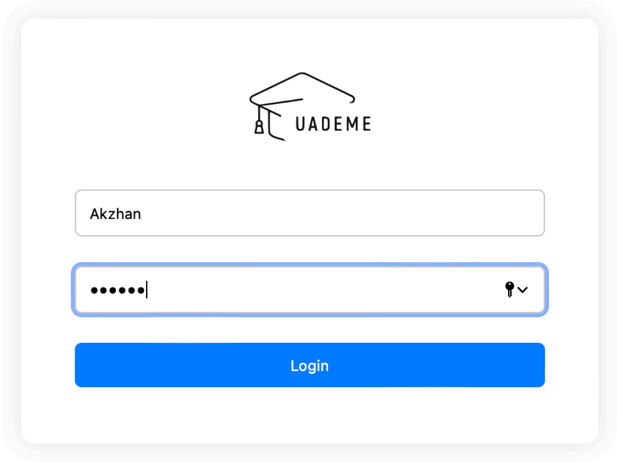
# JWT based authorization

## Token interface

app > TS token.ts > Token

```ts
export interface Token {
  access: string;
  refresh: string;
}
```

## Login Page




## Interceptor

app > TS AuthInterceptor.ts > ...

```ts
import { HTTP_INTERCEPTORS, HttpHandler, HttpInterceptor, HttpRequest } 
import { Injectable } from '@angular/core';

@Injectable()
export class AuthInterceptor implements HttpInterceptor {

  constructor() {
  }

  intercept(req: HttpRequest<any>, next: HttpHandler) {
    const access = localStorage.getItem("access");
    if (access) {
      const newReq = req.clone({
        headers: req.headers.set('Authorization', `Bearer ${access}`)
      });
      return next.handle(newReq);
    }

    return next.handle(req);
  }

}
```

## Service to hit API

app > TS authentication.service.ts > ...

```ts
import {Token} from "./token";

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

  BASE_URL = 'http://localhost:8000/api';
  private logged: boolean = false;

  constructor(private http: HttpClient) {}

  login(username: string, password: string): Observable<Token> {
    return this.http.post<Token>(
      `${this.BASE_URL}/login/`,
      {username, password}
    )
  }

  getLoggedStatus() : boolean {
    return this.logged
  }

  setLoggedStatus(status: boolean) : void {
    this.logged = status
  }
```

## ngModels

```html
<div class="login-container">
  <div class="login-form">
    <div class="logo">
      <img src="assets/images/Logo-2.svg" alt="Logo">
    </div>
    <form action="/home/consultants" method="POST">
      <input [(ngModel)]="username" type="text" name="username" placeholder="Username" required>
      <input [(ngModel)]="password" type="password" name="password" placeholder="Password" required>
      <button (click)="login()" type="submit">Login</button>
    </form>
  </div>
</div>
```

## Services to get Data from API

```ts
export class UniversityService{
  BASE_URl = "http://127.0.0.1:8000/api/universities/"
  constructor(private httpClient: HttpClient) { }

  getUniversities(): Observable<University[]>{
    return this.httpClient.get<University[]>(`${this.BASE_URl}`)
  }

  getUniversity(id: number){
    return this.httpClient.get<University>(`${this.BASE_URl}${id}`)

  }

}
```

# Backend

## Models

```python
class Specialty(models.Model):
    general = models.IntegerField()
    quota = models.IntegerField()
    photo = models.URLField()
    disciplines = models.ManyToManyField(Discipline)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name_plural = 'Specialties'


class University(models.Model):
    name = models.CharField(max_length=100)
    photo = models.URLField()
    specialties = models.ManyToManyField(Specialty)
    phoneNumber = models.CharField(max_length=20)
    cost = models.IntegerField()
    uniType = models.CharField(max_length=100)
    location = models.ForeignKey(City, on_delete=models.CASCADE)
    grantScore = models.IntegerField()
    paidScore = models.IntegerField()
    address = models.TextField()

    def __str__(self):
        return self.name

    class Meta:
        verbose_name_plural = "Universities"


class Consultant(models.Model):
    name = models.CharField(max_length=100)
    profession = models.CharField(max_length=100)
    phone = models.CharField(max_length=20)
    telegram = models.CharField(max_length=100)
```

```python
class TestManager(models.Manager):

    def get_by_test_type(self, test_type):
        return self.filter(test_type=test_type)[0]


class Test(models.Model):

    objects = TestManager()

    TEST_TYPES = [
        ('test1', 'Тест 1'),
        ('test2', 'Тест 2'),
        ('test3', 'Тест 3'),]
    test_type = models.CharField(max_length=20, choices=TEST_TYPES)
    questions = models.TextField()
    variants = models.TextField()
```

## Serializers

```python
class DisciplineSerializer(serializers.ModelSerializer):
    class Meta:
        model = Discipline
        fields = '__all__'


class CitySerializer(serializers.ModelSerializer):
    class Meta:
        model = City
        fields = '__all__'


class SpecialtySerializer(serializers.ModelSerializer):
    class Meta:
        model = Specialty
        fields = '__all__'


class UniversitySerializer(serializers.ModelSerializer):
    class Meta:
        model = University
        fields = '__all__'


class ConsultantSerializer(serializers.ModelSerializer):
    class Meta:
        model = Consultant
        fields = '__all__'
```

## Views, CRUD

```python
@api_view(['GET', 'PUT', 'DELETE'])
def university_detail(request, pk):
    university = get_object_or_404(University, pk=pk)

    if request.method == 'GET':
        serializer = UniversitySerializer(university)
        return Response(serializer.data)

    elif re
        ser                                                    ta=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors, status=400)

    elif request.method == 'DELETE':
        university.delete()
        return Response({'message': 'Discipline deleted successfully'}, status=204)
```

(variable) serializer: UniversitySerializer

```python
class TestListByTypeAPIView(APIView):
    serializer_class = TestSerializer
    permission_classes = [IsAuthenticated]
    def get(self, request, test_type):
        test = Test.objects.get_by_test_type(test_type=test_t
        serializer = self.serializer_class(test)
        return Response(serializer.data)


@api_view(['POST'])
@permission_classes([IsAuthenticated])
def save_test_result(request, test_type, username):
    answers = request.data
    serializer = TestResultSerializer(data={"test_type": test_
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_2
    return Response(serializer.errors, status=status.HTTP_400_


@api_view(['GET'])
@permission_classes([IsAuthenticated])
def get_test_result(request, test_type, username):
    test_results = TestResult.objects.filter(test_type=test_t
    if test_results.exists():
        return Response(test_results.last().answers)
    else:
        return Response({"error": "No test results found for
                         status=status.HTTP_404_NOT_FOUND)
```