Настройка backend

1. Создайте новый проект Django и приложение(в примерах используется **users** для приложения и **auth_backend** для проекта)
2. В users/models.py

```
from django.contrib.auth.models import AbstractUser
from django.db import models

class CustomUser(AbstractUser):
    phone = models.CharField(max_length=20, blank=True)
    bio = models.TextField(max_length=500, blank=True)

    def __str__(self):
        return self.email
```

3. В users/serializers.py

```
from rest_framework import serializers
from django.contrib.auth import get_user_model
from rest_framework.authtoken.models import Token

CustomUser = get_user_model()

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = CustomUser
        fields = ('id', 'username', 'email', 'password', 'phone', 'bio')
        extra_kwargs = {'password': {'write_only': True}}

    def create(self, validated_data):
        user = CustomUser.objects.create_user(**validated_data)
        Token.objects.create(user=user)
        return user

class LoginSerializer(serializers.Serializer):
    username = serializers.CharField()
    password = serializers.CharField()
```

4. users/views.py

```
from rest_framework import generics, permissions, status
from rest_framework.response import Response
from rest_framework.authtoken.models import Token
from django.contrib.auth import authenticate
from .serializers import UserSerializer, LoginSerializer

class RegisterView(generics.CreateAPIView):
    serializer_class = UserSerializer
    permission_classes = [permissions.AllowAny]

class LoginView(generics.GenericAPIView):
    serializer_class = LoginSerializer
    permission_classes = [permissions.AllowAny]

    def post(self, request):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        user = authenticate(**serializer.validated_data)
        if user:
            token, created = Token.objects.get_or_create(user=user)
            return Response({'token': token.key})
        return Response({'error': 'Invalid credentials'},
status=status.HTTP_400_BAD_REQUEST)
```

5. В urls.py

```
from django.contrib import admin
from django.urls import path
from users import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/register/', views.RegisterView.as_view()),
    path('api/login/', views.LoginView.as_view()),
]
```

6. В settings.py

```
UTH_USER_MODEL = 'users.CustomUser'
INSTALLED_APPS = [
    ...,
    'rest_framework',
    'rest_framework.authtoken',
    'corsheaders',
    'users',
]
```

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
    ]
}

CORS_ALLOWED_ORIGINS = [
    "http://localhost:8080",
]

MIDDLEWARE = [
    ...,
    'corsheaders.middleware.CorsMiddleware',
]
```

Настройка frontend

1. Создайте пустой проект vue
2. store/auth.ts

```
import { defineStore } from 'pinia'
import axios from 'axios'

interface User {
  id: number
  username: string
  email: string
  phone?: string
  bio?: string
}

interface AuthState {
  user: User | null
  token: string | null
}

export const useAuthStore = defineStore('auth', {
  state: (): AuthState => ({
    user: null,
    token: localStorage.getItem('token') || null
  }),
  actions: {
    async register(userData: {
      username: string
      email: string
      password: string
      phone?: string
      bio?: string
    }) {
      const response = await axios.post('http://localhost:8000/api/register/', userData)
      this.token = response.data.token
      localStorage.setItem('token', this.token)
      await this.fetchUser()
    },
    async login(credentials: { username: string; password: string }) {
      const response = await axios.post('http://localhost:8000/api/login/', credentials)
      this.token = response.data.token
      localStorage.setItem('token', this.token)
      await this.fetchUser()
    },
    async fetchUser() {
      if (this.token) {
       const response = await axios.get('http://localhost:8000/api/user/', {
        headers: { Authorization: `Token ${this.token}` }
       })
       this.user = response.data
      }
    },
    logout() {
      this.user = null
      this.token = null
      localStorage.removeItem('token')
    }
  }
})
```

3.Компонент Login.vue

```
<script setup lang="ts">
import { ref } from 'vue'
import { useAuthStore } from '@/stores/auth'
import { useRouter } from 'vue-router'

const auth = useAuthStore()
const router = useRouter()
const form = ref({
 username: '',
 password: ''
})
const error = ref('')
```

```
async function submit() {
  try {
    await auth.login(form.value)
    router.push('/dashboard')
  } catch (err) {
    error.value = 'Invalid credentials'
  }
}
</script>

<template>
  <div class="login">
    <h1>Login</h1>
    <form @submit.prevent="submit">
      <input v-model="form.username" placeholder="Username">
      <input v-model="form.password" type="password" placeholder="Password">
      <button type="submit">Login</button>
      <p v-if="error" class="error">{{ error }}</p>
    </form>
  </div>
</template>
```

4. Компонент Dashboard.vue

```
<script setup lang="ts">
import { useAuthStore } from '@/stores/auth'

const auth = useAuthStore()
</script>

<template>
  <div class="dashboard">
    <h1>Welcome {{ auth.user?.username }}</h1>
    <p>Email: {{ auth.user?.email }}</p>
    <p v-if="auth.user?.phone">Phone: {{ auth.user.phone }}</p>
    <p v-if="auth.user?.bio">Bio: {{ auth.user.bio }}</p>
    <button @click="auth.logout()">Logout</button>
  </div>
</template>
```

5. Настройте роутинг в router.ts

```
import { createRouter, createWebHistory } from 'vue-router'
import { useAuthStore } from '@/stores/auth'

const router = createRouter({
  history: createWebHistory(),
  routes: [
    { path: '/', component: () => import('@/views/Home.vue') },
    { path: '/login', component: () => import('@/views/Login.vue') },
    {
      path: '/dashboard',
      component: () => import('@/views/Dashboard.vue'),
      meta: { requiresAuth: true }
    }
  ]
})

router.beforeEach((to, from, next) => {
  const auth = useAuthStore()
  if (to.meta.requiresAuth && !auth.token) {
    next('/login')
  } else {
    next()
  }
})

export default router
```

6. Запустите frontend и backend часть

Дополнительное задание:
Добавить дополнительную страницу, которая будет позволять пользователю редактировать данные о себе.