Specificații Back-End

**Models** (folosite pentru a specifica structura tabelelor din baza de date, folosite la migrări)**:**

**1.User(clasa)**

- sa aibă proprietățile publice:

* id (int),
* firstName (string),
* lastName (string),
* email (string),
* cnp (string),
* dateOfBirth (dateTime),
* phoneNumber (string),
* PasswordHash (byte array),
* PasswordSalt (byte array),
* isApproved (int),
* isMedic (int),
* description (string),
* photo (string),
* doctorId (int),
* Token (string),
* validated (int),
* Pacients (colectie de obiecte User),
* Appointments (colectie de obiecte Appointments);

-o functie Getage() care returnează vârsta (int) unui user.

**2.Appointments(clasa)**

-sa aiba proprietatile publice:

* id(int),
* date (dateTime),
* hour (string),
* userId (int).

**DTOs (data tranfer objects**, tipuri de obiecte primite de API-uri ca input de la client sau returnate de catre API-uri clientului**):**

**1.AccountDTO(clasa)-**

* id (int),
* firstName(string),
* lastName (string),
* email (string),
* phoneNumber(string),
* cnp (string),
* age (int),
* doctorId (int),
* doctor (object MedicDTO);

**2.ChangePasswordDTO(clasa)-**

* oldPassword (string, required),
* newPassword (string, required),
* repeatPassword (sring, verificare daca are aceeasi valoare cu newPassword);

**3. CreateAppointmentDTO(clasa)-**

* date (dateTime, required),
* hour (string, required);

**4. DoctorDTO(clasa)-**

* id (int),
* firstName (string),
* lastName (string),
* email (string),
* phoneNumber (string),
* description (string),
* photo (string);

**5. LoginDTO(clasa)-**

* email (string,required),
* password (string, required);

**6.MedicDTO(clasa)-**

* doctorId (int),
* firstName (string),
* lastName (string),
* phoneNumber (string);

**7.NextOrHistoryAppointmentsDTO(clasa)-**

* id (int),
* date (dateTime),
* hour (string),
* user (object UserDTO)

**8.PatientDTO(clasa)-**

* id(int),
* firstName (string, required, poate contine doar litere mari, litere mici, virgula, apostrof, liniuta si punct),
* lastName (string, required, poate contine doar litere mari, litere mici, virgula, apostrof, liniuta si punct),
* email(string, required, trebuie sa aiba format de email),
* cnp (string, required, trebuie sa fie format din exact 13 cifre),
* age (int),
* phoneNumber (required, trebuie sa aiba formatul unui numar de telefon)

**9.RegisterDTO(clasa)-**

* firstName (string, required, poate contine doar litere mari, litere mici, virgula, apostrof, liniuta si punct),
* lastName (string, required, poate contine doar litere mari, litere mici, virgula, apostrof, liniuta si punct),
* email(string, required, trebuie sa aiba format de email),
* cnp (string, required, trebuie sa fie format din exact 13 cifre),
* phoneNumber (required, trebuie sa aiba formatul unui numar de telefon),
* dateOfBirth ( dateTime, required),
* password(string, required, trebuie sa contina cel putin 8 caractere din care cel putin o litera mare, o litera mica si un numar),
* repeatPassword (string, required, sa se compare egalitatea cu atributul password),
* doctorId(int, required);

**10.ResetPasswordDTO(clasa)-**

* newPassword (string, required, trebuie sa contina cel putin 8 caractere din care cel putin o litera mare, o litera mica si un numar),
* repeatPassword(string, comparare egalitate cu newPassword)
* resetCode (string,required)

**11.ReturnAppointmentsDTO(clasa)-**

* date (dateTime)
* hour (string)

**12.UpdateDoctorDTO(clasa)-**

* firstName (string,required, , poate contine doar litere mari, litere mici, virgula, apostrof, liniuta si punct)
* lastName (string, required, , poate contine doar litere mari, litere mici, virgula, apostrof, liniuta si punct)
* email (string, required, trebuie sa aiba formatul unui email)
* phoneNumber (string, required, trebuie sa aiba formatul unui nr de telefon)
* description (string, required)
* photo (IFormFile)

**13.UpdatePatientDTO(clasa)-**

* firstName (string,required, , poate contine doar litere mari, litere mici, virgula, apostrof, liniuta si punct)
* lastName (string,required, , poate contine doar litere mari, litere mici, virgula, apostrof, liniuta si punct)
* email (string, required, trebuie sa aiba formatul unui email)
* phoneNumber (string, required, trebuie sa aiba formatul unui nr de telefon)

**14.UserDTO(clasa)-**

* id (int)
* email (string)
* firstName (string)
* lastName (string)
* role (int)
* token (string)
* doctorId (int)

**Servicii**(folosite de API-uri):

**1.DatabaseContext(clasa)-** mosteneste din clasa DbContext si reprezinta conexiunea la baza de ate

* USERS (un DbSet de obiecte de tip User)
* APPOINTMENTS (un DbSet de obiecte de tip Appointments)

**2. TokenService(clasa)-** conține metoda CreateToken(User user): crează un token pentru user-ul primit în input, returnează token-ul creat de tip string

**Controllers:**

**1.UsersController(clasa)**: contine toate API-urile necesare care sunt legate de tabela users, are ca si ruta “api/[controller]”

User APIs(metode):

1. getPatients()

-ruta: “getPatients”

-utilizare: returneaza detaliile tuturor pacientilor unui medic logat

-input: -nimic-

-output: lista de obiecte PatientDTO

-autorizare: doar pentru medici logati

1. getDoctors()

-ruta: “getDoctors”

-utilizare: returneaza detaliile tuturor doctorilor din baza de date

-input: -nimic-

-output: o lista de obiecte DoctorDTO

-autorizare: pentru toti utilizatorii logati cat si nelogati

1. getUser()

-ruta: ”MyAccount”

-utilizare: returneaza informatii despre user-ul logat( utilizat pentru pagina MyAccount a userului logat)

-input: -nimic-

-output: un obiect de tipul AccountDTO

-autorizare: doar pentru pacienti logati

1. getDoctor()

-ruta: “MyAccountMedic”

-utilizare: returneaza informatii despre medical logat (folosit pentru pagina de myAccount a medicului)

-input: -nimic-

-output: un obiect de tipul DoctorDTO

-autorizare: doar pentru medici logati

1. getPatientInfo(int id)

-ruta: “getPatientInfo”

-utilizare: returneaza informatii despre un pacient al unui doctor logat

-input: int id (id-ul pacientului despre care se doresc informatii)

-output: un obiect de tipul PatientDTO

-autorizare: doar pentru medici logati

1. getDoctorInfo(int id)

-ruta: "getMedicInfo"

-utilizare: returneaza informatii unui medic selectat de utilizator

-input: int id (id-ul medicului despre care se doresc informatii)

-output: un obiect de tipul DoctorDTO

-autorizare: pentru toti utilizatorii logati si nelogati

1. deletePatient(int id)

-ruta: "DeletePatient"

-utilizare: sterge din baza de date pacientul selectat de catre un medic

-input: int id (id-ul pacientului selectat de catre medic)

-output: -nimic-

-autorizare: doar pentru medici logati

1. deleteAccount()

-ruta: "DeleteAccount"

-utilizare: sterge user-ul logat din baza de date (user-ul isi poate sterge contul din MyAccount)

-input: -nimic-

-output: -nimic-

-autorizare: pentru orice user logat

1. getUnapprovedUsers()

-ruta: "getUnapprovedUsers"

-utilizare: returneaza pacientii inca neaprobati ai unui medic logat

-input: -nimic-

-output: o lista de obiecte de tipul PatientDTO

-autorizare: doar pentru medici logati

1. ApproveUser(int id)

-ruta: "ApproveUser"

-utilizare: aproba contul unui pacient al doctorului logat (schimba isApproved= 0 in isApproved=1 in baza de date)

-input: int id (id-ul pacientului care se doreste sa fie aprobat)

-output: -nimic-

-autorizare: doar pentru medici logati

1. UpdateUser(UpdatePatientDTO patientDTO)

-ruta: "updateUser"

-utilizare: actualizeaza informatiile din baza de date ale pacientului logat cu informatiile trimise de acesta prin formularul din pagina MyAccount

-input: un obiect de tipul UpdatePatientDTO

-output: -nimic-

-autorizare: doar pentru pacienti logati

1. UpdateMedic(UpdateDoctorDTO doctorDTO)

-ruta: "updateMedic"

-utilizare: actualizeaza informatiile unui medic logat din baza de date (pagina MyAccount a medicului)

-input: un obiect de tipul UpdateDoctorDTO

-output: -nimic-

-autorizare: doar pentru medici logati

1. changePassword(ChangePasswordDTO changePasswordDTO)

-ruta: "ChangePassword"

-utilizare: actualizeaza in baza de date parola utilizatorului logat

-input: obiect de tipul ChangePasswordDTO

-output: -nimic-

-autorizare: pentru toti utilizatorii logati

1. searchUser(string str)

-ruta: "searchUser"

-utilizare: pentru functionalitatea de cautare a pacientilor ai unui medic logat, returneaza pacientii medicului logat a caror nume, prenume sau nume si prenume concatenate se potriveste cu string-ul transmis ca si input

-input: string str

-output: o lista de obiecte de tipul PatientDTO

-autorizare: doar pentru medici logati

1. searchDoctor(string str)

-ruta: "searchDoctor"

-utilizare: cauta in baza de date doctorii ai caror nume, prenume, concatenate nume-prenume se potriveste cu string-ul transmis ca si parametru

-input: string str

-output: o lista de obiecte de tipul DoctorDTO

-autorizare: pentru toti utilizatorii logati si nelogati

1. Register(RegisterDTO RegisterDTO)

-ruta: "register"

-utilizare: pentru inregistrare unui user nou in baza de date

-input: un obiect de tipul RegisterDTO

-output: un obiect de tipul User

-autorizare: pentru toti utilizatorii logati si nelogati

1. Login(LoginDTO LoginDto)

-ruta: "login"

-utilizare: pentru logarea unui utilizator

-input: un obict de tipul LoginDTO

-output: un obiect de tipul UserDTO

-autorizare: pentru toti utilizatorii logati si nelogati

1. ResetPassword(ResetPasswordDTO model)

-ruta: "ResetPassword"

-utilizare: actualizeaza in baza de date parola unui utilizator pe baza unui token unic

-input: un obiect de tipul ResetPassowrdDto

-output: -nimic-

-autorizare: pentru toti utilizatorii logati si nelogati

1. VerifyAccount(string token)

-ruta: "VerifyAccount"

-utilizare: verifica un cont (schimba atributul verified a unui cont din baza de date din 0 in 1)

-input: string token

-output: -nimic-

-autorizare: pentru toti userii loati si nelogati

Metode ajutătoare:

1. UserExists(string email) – verifica daca exista un user in baza de date cu emai-ul dat ca parametru
2. SendEmail(string emailAddress, string body, string subject) – trimite un email cu subiectul si continutul transmise ca si parametrii adresei de email trimisa ca parametru (email trimis de pe adresa de email a clinicii)
3. EmailVerification(string email) – se trimite un email catre adresa transmisa ca si parametru cu un token unic de ferificare.

**2.AppointmentsController(clasa)**: contine toate API-urile necesare care sunt legate de tabela appointments are ca si ruta “api/[controller]”

Appointment APIs(metode):

1. createAppointment(CreateAppointmentDTO app)

-ruta: "createApp"

-utilizare: creaza o noua programare in baza de date

-input: un obiect de tipul CreateAppointmentDTO

-output: un obiect de tipul ReturnAppointmentsDTO

-autorizare: doar pentru utilizatorii logati

1. getAppointments(int userId)

-ruta: "{userId}"

-utilizare: returneaza programarile user-ului logat

-input: int userId

-output: o lista de obiecte Appointments

-autorizare: doar pentru pacientii logati

1. getBackApp()

-ruta: "historyAppointments"

-utilizare: returneaza toate programarile unui user care au data mai mica decat data curenta

-input: -nimic-

-output: o lista de obiecte de tipul NextOrHistoryAppointmentsDTO

-autorizare: doar pentru pacientii logati

1. getNextApp()

-ruta: "nextAppointments"

-utilizare: returneaza programarile utilizatorului logat care au data mai mare decat data curenta

-input: -nimic-

-output: o lista de obiecte de tipul NextOrHistoryAppointmentsDTO

-autorizare: doar pentru pacientii logati

1. getBackAppByMedic(int Id)

-ruta: "historyAppointmentsByMedic"

-utilizare: returneaza programarile tuturor pacientilor medicului logat care au data mai mica decat data curenta

-input: int Id (id-ul doctorului logat)

-output: o lista de obiecte de tipul NextOrHistoryAppointmentsDTO

-autorizare: doar pentru doctori logati

1. getNextAppByMedic(int Id)

-ruta: "nextAppointmentsByMedic"

-utilizare: returneaza toate programarile pacientilor unui medic logat care au data mai mare decat data curenta

-input: int ID (id-ul medicului logat)

-output:o lista de obiecte de tipul NextOrHistoryAppointmentsDTO

-autorizare: doar pentru medici logati

1. getAllUsers(int id)

-ruta: "allDoctorApp/{id}"

-utilizare: returneaza toate programarile unui medic logat

-input:int id (id-ul medicului logat)

-output: o lista de obiecte cu proprietatile (MedicId (int, id-ul medicului logat) pactientFirstName (string), pacientLastName (string), phone (string), email (string), DateOfApp (dateTime), HourOfApp(string))

-autorizare: doar pentru medici logati

1. deleteApp(int appId)

-ruta: "delete/{appId}"

-utilizare: sterge o programare selectata de utilizator din baza de date

-input: int appId (id-ul programarii selectate)

-output: -nimic-

-autorizare: pentru toti utilizatorii logati

1. getAppByDate(DateTime date, int id)

-ruta: "getMyAppointments"

-utilizare: returneaza toate programarile dintr-o anumita zi ale unui medicului logat

-input: dateTime date (ziua selectata), int id (id-ul medicului logat)

-output: o lista de obieccte de tipul NextOrHistoryAppointmentsDTO

-autorizare: pentru toti utilizatorii logati

Metode ajutătoare:

1. AppointmentDateExist(DateTime date)- ferifica daca exista o programare in baza de date la data trimisa ca si parametru

2. AppointmentHourExist(string Hour)- ferifica daca exista in baza de date vreo programare la ora transisa ca si parametru