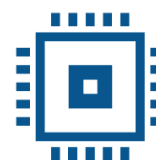




Universitatea *Transilvania* din Braşov
Facultatea de Inginerie Electrică şi Ştiinţa Calculatoarelor
Departamentul Automatică şi Tehnologia Informaţiei



PROIECT AGENDĂ SPRING BOOT

Profesor îndrumător: Conferenţiar universitar ,
KRISTÁLY Dominic Mircea

Grabovenco Bogdan-Iulian

BRAŞOV, 2024

Cuprins

1.1.	Enunt.....	2
1.2.	Rezolvare Proiect	2
1.3.	Baza de date	2
1.4.	Backend	7
1.5.	POSTMAN	23
1.6.	Frontend	25
1.7.	Manual de utilizare	36

1. Agendă Spring Boot

1.1. Enunt

Să se realizeze o aplicație care să implementeze toate funcțiile necesare unei agende personale. Aplicația va oferi modulele:

1. Autentificare
2. Vizualizare note
3. Deconectare

Aplicația poate conține orice alte module care sunt considerate a fi necesare.

1.2. Rezolvare Proiect

Prezentarea proiectului se va face etapizat în ordinea în care s-au abordat problemele, și anume ordinea ierarhizată care începe cu modelarea bazei de date, apoi stabilirea unei legături între baza de date și mediul de lucru IntelliJ, unde s-a lucrat cu framework-ul Spring Boot și limbajul Java, iar la final stabilirea legăturii externe, conectarea prin intermediul framework-ului de frontend React, realizându-se un API simplu care permite utilizatorului să interacționeze mascat prin metode intermediare cu baza de date. Cu alte cuvinte, prima parte este stocarea, a doua administrarea acesteia prin introducerea de reguli la nivel backend, iar a treia și ultima parte implementarea unei interfețe între utilizator și mediul de stocare.

1.3. Baza de date

Baza de date este modelată cu ajutorul MySQL, aplicația dezvoltată de ORACLE, s-a ales lucrul cu această aplicație datorită accesibilității, preferându-se în detrimentul MySQL de la Microsoft din cauza problemelor întâmpinate în stabilirea legăturii dintre mediul de stocare al bazei de date în sine și softul care nu permitea stabilirea unei legăturii fără a se stabili în prealabil o metodă de conectare la baza de date în criterii de siguranță. Varianta celor de la ORACLE impune stabilirea unei metode sigure de autentificare și acces la baza de date în mod implicit, iar setările care decurg din aceasta sunt manipulate de aplicație. În continuare o să se prezinte interogările, care sunt în număr de 3, cu care s-a creat baza de date și entitățile cuprinse în aceasta, precum și modul de relaționare între entități, împreună cu constrângerile asociate atributelor. În plus sunt inserate și câteva date pentru scopuri de prelucrare și demonstrație. Interogările conțin și erori de sintaxă care au fost menținute pentru lucru comparativ cu sintaxa celor de la Microsoft care diferă prin câteva aspecte minore.

Primul script:

BD Secvență de cod 1

```
show databases;
create database DB1;
use DB1;use sys;drop database DB1;
create table Tab1
(
ID int primary key identity,
Name nvarchar(100),
Age int
);
insert into Tab1 values ('Ion',21),('Dorin',12);
select * from Tab1;

create database Accountt;
use Accountt;
create table Userr
(
ID int not null auto_increment primary key,
Username nvarchar(50) not null,
Passwordd nvarchar(50) not null
);
insert into Userr (Username, Passwordd) values
('CojocDeUrs', 'mimi2'),
('FeciorulDeAur', 'tup5');
select * from Userr;
show tables;
drop table Persoana;
select * from persOana;
/* new start 01.04 with Arjun*/
use Accountt;
select * from Student;
select * from Userr;
select * from User;
/*03.04*/
use Accountt;
create table Masina
(
ID int auto_increment primary key,
Marca nvarchar(50),
Model nvarchar(30),
Culoare nvarchar(25)
);
insert into Masina (Marca, Model, Culoare) values
('VW', 'Golf 3', 'alb'),
('BMW', 'M3', 'rosu'),
('Dacja', 'Logan', 'verde');
select * from Masina;

/*06.04*/
use sys;
drop database Accountt;
drop database DiaryUser;

create database DiaryUser character set latin1 collate latin1_general_cs;
use DiaryUser;
```

```
create table `User`
(
  `ID` int auto_increment primary key,
  `Username` varchar(35) not null,
  `Password` varchar(25) not null,
  `Email` varchar(38) not null,
  `Type` varchar(8) default 'Standard',
  constraint `User_c1` check (`Type` = 'Standard' or `Type` = 'Premium')
);
alter table `User` add unique (`Username`);
alter table `User` add unique (`Email`);
alter table `User` add constraint `User_c2` check (char_length(`Password`)
>= 8);

insert into `User` (`Username`, `Password`, `Email`, `Type`) values
('Stefan32', 'cod-trecere-cod-trecere', 'stefan2001_21@yahoo.ro',
'Standard'),
('Ionel', 'bustean-voluminos123', 'ionel_orasanu38@yahoo.com', 'Premium'),
('Raul Pop', 'gazonVerdeTOP', 'player_RAOL@gmail.com', 'Standard'),
('Anton Cocor', 'balastiera321', 'cocor_de_apaDulce2005@gmail.com',
'Standard');

select * from `User`;

create table `SecurityQuestion`
(
  `ID` int auto_increment primary key,
  `Question` varchar(40) not null,
  `Answer` varchar(15) not null,
  `ID_User` int,
  constraint `SecurityQuestion_fk` foreign key (`ID_User`) references
  `User`(`ID`),
  constraint `SecurityQuestion_c1` check (
  `Question` = 'What is your dog name?' or
  `Question` = 'What is your childhood nickname?' or
  `Question` = 'When did you finish school?' or
  `Question` = 'Where did you grow up?' or
  `Question` = 'What sport are most attracted to?'),
  constraint `SecurityQuestion_c2` check (char_length(`Answer`) >= 5)
);

insert into `SecurityQuestion` (`Question`, `Answer`, `ID_User`) values
('Where did you grow up?', 'Oslo City', 1),
('What is your dog name?', 'Rufus', 2),
('Where did you grow up?', 'Helsinki', 3),
('What sport are most attracted to?', 'Cricket', 4);

select * from `SecurityQuestion`;

create table `Agenda`
(
  `ID` int auto_increment primary key,
  `Notes` varchar(65000),
  `Date` datetime default current_timestamp,
  `ID_User` int,
  constraint `Agenda_fk` foreign key (`ID_User`) references `User` (`ID`)
);

insert into `Agenda` (`Notes`, `ID_User`) values
('uda florile cu ingrasamant', 1);
```

```
insert into `Agenda` (`Notes`, `ID_User`) values
('suna pe Mariana pentru programare dentist!!', 1);
insert into `Agenda` (`Notes`, `ID_User`) values
('rescrie recenzie carte', 1);
insert into `Agenda` (`Notes`, `ID_User`) values
('program sala - mult sport', 3);
insert into `Agenda` (`Notes`, `ID_User`) values
('de vizitat parcul Brandusa!, sun-o pe Dorina', 2);

select * from `Agenda`;
```

Al doilea script:

BD Secvență de cod 2

```
use sys;
drop database DiaryUser;

create database DiaryUser character set latin1 collate latin1_general_cs;
use DiaryUser;

create table `User`
(
  `ID` int auto_increment primary key,
  `Username` varchar(35) not null,
  `Password` varchar(25) not null,
  `Email` varchar(38) not null,
  `Type` varchar(8) default 'Standard',
  constraint `User_c1` check (`Type` = 'Standard' or `Type` = 'Premium')
);
alter table `User` add unique (`Username`);
alter table `User` add unique (`Email`);
alter table `User` add constraint `User_c2` check (char_length(`Password`)
>= 8);

insert into `User` (`Username`, `Password`, `Email`, `Type`) values
('Stefan32', 'cod-trecere-cod-trecere', 'stefan2001_21@yahoo.ro',
'Standard'),
('Ionel', 'bustean-voluminos123', 'ionel_orasanu38@yahoo.com', 'Premium'),
('Raul Pop', 'gazonVerdeTOP', 'player_RAOL@gmail.com', 'Standard'),
('Anton Cocor', 'balastiera321', 'cocor_de_apaDulce2005@gmail.com',
'Standard');

select * from `User`;
select distinct Type from `User`;

create table `SecurityQuestion`
(
  `ID` int auto_increment primary key,
  `Question` varchar(40) not null,
  `Answer` varchar(15) not null,
  `ID_User` int,
  constraint `SecurityQuestion_fk` foreign key (`ID_User`) references
  `User`(`ID`),
  constraint `SecurityQuestion_c1` check (
    `Question` = 'What is your dog name?' or
    `Question` = 'What is your childhood nickname?' or
    `Question` = 'When did you finish school?' or
```

```
`Question` = 'Where did you grow up?' or
`Question` = 'What sport are most attracted to?'),
constraint `SecurityQuestion_c2` check (char_length(`Answer`) >= 5)
);

insert into `SecurityQuestion` (`Question`, `Answer`, `ID_User`) values
('Where did you grow up?', 'Oslo City', 1),
('What is your dog name?', 'Rufus', 2),
('Where did you grow up?', 'Helsinki', 3),
('What sport are most attracted to?', 'Cricket', 4);

select * from `SecurityQuestion`;

create table `Agenda`
(
`ID` int auto_increment primary key,
`Notes` varchar(65000),
`Date` datetime default current_timestamp,
`ID_User` int,
constraint `Agenda_fk` foreign key (`ID_User`) references `User` (`ID`)
);

update `Agenda` set ID_User = 1 where ID_User is null;
alter table `Agenda` modify `ID_User` int not null;

insert into `Agenda` (`Notes`, `ID_User`) values
('uda florile cu ingrasamant', 1);
insert into `Agenda` (`Notes`, `ID_User`) values
('suna pe Mariana pentru programare dentist!!!', 1);
insert into `Agenda` (`Notes`, `ID_User`) values
('rescrie recenzie carte', 1);
insert into `Agenda` (`Notes`, `ID_User`) values
('program sala - mult sport', 3);
insert into `Agenda` (`Notes`, `ID_User`) values
('de vizitat parcul Brandusa!, sun-o pe Dorina', 2);
insert into `Agenda` (`Notes`, `ID_User`) values
('mananca sanatos', 3);

select * from `Agenda`;

/* it works this way */
insert into `Agenda` (`ID_User`, `Notes`) values
(
(select `ID` from `User` where `Username` = 'Doru Dorut'),
'notita buna!!!'
);

select Notes from Agenda where ID_User = (select ID from User where Username
= 'Stefan32');
select Notes from Agenda where ID_User = (select ID from User where Username
= 'Stefan32');
```

Al treilea și ultimul script:

BD Secvență de cod 3

```
use DiaryUser;
```

```
show tables;
show databases;
drop table student;
drop table user_new;
drop table persoana;
```

1.4. Backend

Backend-ul are rolul de-a juca un rol de intermediar între baza de date și API, interfața utilizatorului, acesta lansând request-uri (cereri) către baza de date. Cererile sunt de tipul GET, POST, PUT, care corespund tipurilor de interogări elementare ce se pot efectua în bazele de date. Interogările elementare se mai și CRUD, create, read, update și delete. Programul definit comunică aceste cereri serverului, care, la rândul-i, oferă un răspuns bazat pe parametri care pleacă odata cu request-ul din partea de backend. GET acoperă partea de read, POST tot partea de read, dar cu parametri transmiși prin JSON, javascript object notation, dar se pot folosi și alte metode de comunicare de parametri. Apoi PUT corespunde lui create. DEL, care în acest proiect nu a cunoscut utilizare, corespunde eponimului delete. Cu POST se poate realiza și update, dar și aceasta nu a fost folosită în cadrul proiectului.

O mențiune importantă este cum s-a abordat maparea pe entitățile ce se regăses în baza de date. O să se descrie în continuare etapele de realizare a acestui demers.

În primul rând, s-au create clasele ce se pliază exact pe entitățile bazei de date, fiecare atribut corespunzând unei variabile. Fiecare clasă, de-asemeni, corespunde modelului Java de creare a unei clase, ceea ce implică getter-ii, setter-ii, constructorul și metoda toString.

Backend Secvență de cod 1 Utilizatorul

```
package com.daspaket.diary.model;

import jakarta.persistence.*;
import org.springframework.web.bind.annotation.CrossOrigin;

@Entity
@CrossOrigin("http://localhost:3001") //or 3000
@Table(name = "User") //imi asigura ca nu-mi creeaza un alt tabel + sa
scriem ca faca doar update in app.prop
public class User
{
    //
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private int ID;
    @Column(name = "Username")
    private String Username;
    @Column(name = "Password")
    private String Password;
    @Column(name = "Email")
    private String Email;
    @Column(name = "Type")
    private String Type;
```



```
public User()
{
    //default constructor
}

public int getID() {
    return ID;
}

public void setID(int ID1) {
    this.ID = ID1;
}

public String getUsername() {
    return Username;
}

public void setUsername(String username1) {
    Username = username1;
}

public String getPassword() {
    return Password;
}

public void setPassword(String password1) {
    Password = password1;
}

public String getEmail() {
    return Email;
}

public void setEmail(String email1) {
    Email = email1;
}

public String getType() {
    return Type;
}

public void setType(String type1) {
    Type = type1;
}

@Override
public String toString() {
    return "User{" +
        "ID=" + ID +
        ", Username='" + Username + '\'' +
        ", Password='" + Password + '\'' +
        ", Email='" + Email + '\'' +
        ", Type='" + Type + '\'' +
        '}';
}
}
```

```
package com.daspaket.diary.model;

import jakarta.persistence.*;
import org.springframework.web.bind.annotation.CrossOrigin;

@Entity
//@CrossOrigin("http://localhost:3001") //or 3000
@Table(name = "SecurityQuestion")
public class SecurityQuestion
{
    //
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private int ID;
    @Column(name = "Question")
    private String Question;
    @Column(name = "Answer")
    private String Answer;
    @Column(name = "ID_User")
    private int ID_User;

    public SecurityQuestion()
    {
        //default constructor
    }

    public int getID() {
        return ID;
    }

    public void setID(int ID1) {
        this.ID = ID1;
    }

    public String getQuestion() {
        return Question;
    }

    public void setQuestion(String question1) {
        Question = question1;
    }

    public String getAnswer() {
        return Answer;
    }

    public void setAnswer(String answer1) {
        Answer = answer1;
    }

    public int getID_User() {
        return ID_User;
    }

    public void setID_User(int ID_User1) {
        this.ID_User = ID_User1;
    }
}
```

```
    }

    @Override
    public String toString() {
        return "SecurityQuestion{" +
            "ID=" + ID +
            ", Question='" + Question + '\'' +
            ", Answer='" + Answer + '\'' +
            ", ID_User=" + ID_User +
            '}';
    }
}
```

Backend Secvență de cod 3 Agenda

```
package com.daspaket.diary.model;

import jakarta.persistence.*;
import java.util.Date;

@Entity
//CORS
@Table(name = "Agenda")
public class Agenda
{
    //
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private int ID;
    @Column(name = "Notes")
    private String Notes;
    @Column(name = "Date")
    private Date Date;
    @Column(name = "ID_User")
    private int ID_User;

    public Agenda()
    {
        //default constructor
    }

    public int getID() {
        return ID;
    }

    public void setID(int ID1) {
        this.ID = ID1;
    }

    public String getNotes() {
        return Notes;
    }

    public void setNotes(String notes1) {
        Notes = notes1;
    }

    public java.util.Date getDate() {
```

```
        return Date;
    }

    public void setDate(java.util.Date date1) {
        Date = date1;
    }

    public int getID_User() {
        return ID_User;
    }

    public void setID_User(int ID_User1) {
        this.ID_User = ID_User1;
    }

    @Override
    public String toString() {
        return "Agenda{" +
            "ID=" + ID +
            ", Notes='" + Notes + '\'' +
            ", Date=" + Date +
            ", ID_User=" + ID_User +
            '\'';
    }
}
```

Apoi urmează și celelalte obiecte ce au fost folosite pentru diverse metode unde nu era permis utilizarea de date primitive. Spre exemplu, s-a folosit un obiect denumit Stringulescu pentru a putea transforma tipul de dată String, care nu este primitiv ca int sau char, dar este tratat la fel în ce privește metodele folosite în cursul programului. Acest Stringulescu conține doar o variabilă de tip String, la fel s-a procedat pentru un alt obiect unde a fost necesară de crearea unui obiect redundant, dar necesar pentru a folosi metoda respectivă pusă la îndemână de clasa externă/sursa externă Jpa Repository. Din acest punct de vedere, și acestea se încadrează la pachetul model.

Backend Secvență de cod 4 DataTransferObject

```
package com.daspaket.diary.model;

public class DataTransferObject {
    private String username;
    private String notes;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username1) {
        this.username = username1;
    }

    public String getNotes() {
        return notes;
    }

    public void setNotes(String notes1) {
        this.notes = notes1;
    }
}
```

```
}  
}
```

Backend Secvență de cod 5 Stringulescu

```
package com.daspaket.diary.model;  
  
public class Stringulescu {  
    private String username;  
  
    public String getUsername() {  
        return username;  
    }  
  
    public void setS(String s1) {  
        this.username = s1;  
    }  
}
```

În al doilea rând, pentru lucrul explicit cu baze de date s-a folosit un tool excelent care este definit de Jpa Repository, care independent de acesta s-au creat interogări personalizate acolo unde a fost nevoie. Fiecare clasă, adică fiecare clasă din pachetul model, are un corespondent de Repository.

Backend Secvență de cod 6 Agenda Repository

```
package com.daspaket.diary.repository;  
  
import com.daspaket.diary.model.Agenda;  
import com.daspaket.diary.model.DataTransferObject;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Modifying;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface AgendaRepository extends JpaRepository<Agenda, Integer>  
{  
    @Modifying  
    @Transactional  
    @Query(value = "insert into Agenda(ID_User, Notes) " +  
        "values ((select ID from User where Username = :username),  
:notes)", nativeQuery = true)  
    void addPageRepo(@Param("username") String username, @Param("notes")  
String notes);  
  
    @Query("select ID_User, Notes from Agenda where ID_User = (select ID  
from User where Username = :username) and Notes = :notes")  
    Optional<DataTransferObject> checkPageRepo(String username, String  
notes);  
  
    @Query("select Notes from Agenda where ID_User = (select ID from User  
where Username = :username)")  
    List<String> showNotesRepo(String username);  
}
```

```
}
```

Backend Secvență de cod 6 Agenda Repository

```
package com.daspaket.diary.repository;

import com.daspaket.diary.model.Agenda;
import com.daspaket.diary.model.DataTransferObject;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.Optional;

public interface AgendaRepository extends JpaRepository<Agenda, Integer>
{
    @Modifying
    @Transactional
    @Query(value = "insert into Agenda(ID_User, Notes) " +
        "values ((select ID from User where Username = :username),
:notes)", nativeQuery = true)
    void addPageRepo(@Param("username") String username, @Param("notes")
String notes);

    @Query("select ID_User, Notes from Agenda where ID_User = (select ID
from User where Username = :username) and Notes = :notes")
    Optional<DataTransferObject> checkPageRepo(String username, String
notes);

    @Query("select Notes from Agenda where ID_User = (select ID from User
where Username = :username)")
    List<String> showNotesRepo(String username);
}
```

Backend Secvență de cod 7 User Repository

```
package com.daspaket.diary.repository;

import com.daspaket.diary.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;

@Repository
public interface UserNewRepository extends JpaRepository<User, Integer>
{
    /*
    This is an annotation provided by Spring Data JPA that allows you to
    define custom
    JPQL (Java Persistence Query Language) queries directly in your repository
    interface.
    SELECT u: This part specifies that you want to select entities of type
    User.
    */
}
```

```
In JPQL, u is an alias for the User entity.FROM User u: This part
specifies the
entity type (User) and gives it the alias u. It's saying that you're
selecting entities
from the User table and referring to each entity as u.
*/
@Query("select u from User u where u.Username = :in_name")
Optional<User> findByNameRepo(String in_name); //:name must be name
//not useful when attempting a login process

@Query("select u from User u where u.Username = :username and u.Password
= :password")
Optional<User> loginProcessRepo(String username, String password);

@Query("select u from User u where u.Username = :username and u.Password
= :password and u.Email = :email and u.Type = :type")
Optional<User> findUserByUserNewRepo(String username, String password,
String email, String type);

//folositi cand afisam tipul de user cand creem un cont, afisare dinamic
preferabila in locul celei hardcodate
@Query("select distinct Type from User")
List<String> displayTypeRepo();
}
```

Backend Secvență de cod 8 Security Repository

```
package com.daspaket.diary.repository;

import com.daspaket.diary.model.SecurityQuestion;
import org.springframework.data.jpa.repository.JpaRepository;

public interface SecurityQuestionRepository extends
JpaRepository<SecurityQuestion, Integer>
{
    //
}
```

În al treilea rând, s-a stabilit cupletul Service și ServiceImplementation, unde fiecare model are un cuplet. Service este de tipul interfață, iar ServiceImplementation face implementarea metodei declarate în Service. Acest procedeu este utilizat pentru separarea declarării de implementare, mai departe în program inițializarea are loc pe baza lui Service, iar trimiterea se va face către Service, nu către ServiceImplementation. Cupletul are rol de încapsulare, de ascunderea a cât mai multe date atunci când lucrul devine laborios, iar încărcarea cu cât mai multe variabile încarcă ecranul și micșorează vizibilitatea, dar acest lucru este cu mai evident în cazurile unde nu se dorește vizualizarea componentelor, încapsularea jucând rol de mască. Fiecare model are deci un Service și un ServiceImplementation asociat. Acest demers are rolul de-a degaja diverse componente pentru a dezbăra programul principal, astfel crește vizibilitatea și controlabilitatea totodată.

Backend Secvență de cod 9 Agenda Service

```
package com.daspaket.diary.service;

import com.daspaket.diary.model.Agenda;
import com.daspaket.diary.model.DataTransferObject;

import java.util.List;
import java.util.Optional;

public interface AgendaService {
    //
    public List<Agenda> getAllNotes();
    public void addPage(String username, String notes);
    public DataTransferObject checkPage(String username, String notes);
    public List<String> showNotes(String username);
}
```

Backend Secvență de cod 10 Agenda Implementare

```
package com.daspaket.diary.service;

import com.daspaket.diary.model.Agenda;
import com.daspaket.diary.model.DataTransferObject;
import com.daspaket.diary.repository.AgendaRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class AgendaServiceImpl implements AgendaService {
    //
    @Autowired
    AgendaRepository repo;

    @Override
    public List<Agenda> getAllNotes() {
        return repo.findAll();
    }

    public void addPage(String username, String notes)
    {
        //
        repo.addPageRepo(username, notes);
        /*if (list.isPresent())
        {
            //
            return list.get();
        }
        else
        {
            //
            return null;
        }*/
    }
    public DataTransferObject checkPage(String username, String notes)
```



```
{
    //
    Optional<DataTransferObject> list = repo.checkPageRepo(username,
notes);
    if (list.isPresent())
    {
        //
        return list.get();
    }
    else
    {
        //
        return null;
    }
}
public List<String> showNotes(String username)
{
    //
    List<String> list = repo.showNotesRepo(username);
    if(!list.isEmpty())
    {
        return list;
    }
    else
    {
        return null;
    }
}
}
```

Backend Secvență de cod 11 User Service

```
package com.daspaket.diary.service;

import com.daspaket.diary.model.User;

import java.util.List;

public interface UserService
{
    //
    //declaram signatura functiilor pe care le explicitam in UserServiceImpl
    public List<User> getAllUsers();
    public User findUserById(int id);
    public User findUserByUsername(String username);
    public User findUserByUserNew(User user);
    public User loginProcess(String username, String password);
    public User addUser(User user);
    public List<String> displayType();
}
```

```
package com.daspaket.diary.service;

import com.daspaket.diary.model.User;
import com.daspaket.diary.repository.UserNewRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class UserServiceImpl implements UserService
{
    //
    @Autowired
    private UserNewRepository repo;
    @Override
    public List<User> getAllUsers()
    {
        //
        return repo.findAll();
    }
    @Override
    public User findUserById(int id)
    {
        //
        return repo.findById(id).get();
    }
    @Override
    public User findUserByUsername(String username)
    {
        Optional<User> userOptional = repo.findByNameRepo(username);
        if(userOptional.isPresent())
        {
            //User user = userOptional.get();
            return userOptional.get();
        }
        else
        {
            return null; // Handle null if the user doesn't exist
        }
        //return userOptional; fail
        //return repo.findByNameRepo(username);
    }
    public User loginProcess(String username, String password)
    {
        //
        Optional<User> userOptional = repo.loginProcessRepo(username,
password);
        if(userOptional.isPresent())
        {
            //User user = userOptional.get(); return user;
            return userOptional.get();
        }
        else
        {
            return null;
        }
    }
}
```

```
}
public User addUser(User user)
{
    return repo.save(user);
}
public User findUserByUserNew(User user)
{
    //
    Optional<User> userOptional =
repo.findUserByUserNewRepo(user.getUsername(), user.getPassword(),
user.getEmail(), user.getType());
    if(userOptional.isPresent())
    {
        return userOptional.get();//conversie din tipul Optional in
tipul modelului meu, User
    }
    else
    {
        return null;
    }
}
public List<String> displayType()
{
    //
    List<String> StringList = repo.displayTypeRepo();
    if(!StringList.isEmpty())
    {
        return StringList;//
    }
    else
    {
        return null;
    }
}
}
```

Backend Secvență de cod 13 Question Service

```
package com.daspaket.diary.service;

import com.daspaket.diary.model.SecurityQuestion;

import java.util.List;

public interface SecurityQuestionService
{
    //
    public List<SecurityQuestion> getAllQuestions();
}
}}
```

Backend Secvență de cod 14 Question Implementare

```
package com.daspaket.diary.service;

import com.daspaket.diary.model.SecurityQuestion;
import com.daspaket.diary.repository.SecurityQuestionRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class SecurityQuestionServiceImpl implements SecurityQuestionService
{
    //
    @Autowired
    private SecurityQuestionRepository repo;
    @Override
    public List<SecurityQuestion> getAllQuestions()
    {
        //
        return repo.findAll();
    }
}
```

În al patrulea rând, se vor stabili controlere caracteristice fiecărui model. Acestea apelează Service-ul stabilit anterior, iar adițional returnează un mesaj de control care are rol de verificare a funcționalității din Service. Tot aici se stabilesc metodele de comunicare așa cum s-au menționat mai devreme corespondente tipologiei CRUD.

Backend Secvență de cod 15 Agenda Controller

```
package com.daspaket.diary.controller;

import com.daspaket.diary.model.Agenda;
import com.daspaket.diary.model.DataTransferObject;
import com.daspaket.diary.model.Stringulescu;
import com.daspaket.diary.service.AgendaService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping(path = "/agenda")
public class AgendaController
{
    @Autowired
    AgendaService service;
    @GetMapping(path = "/all")
    public List<Agenda> showAll()
    {
        return service.getAllNotes();
    }
    @PutMapping(path = "/add-note")
    public void insertPage(@RequestBody DataTransferObject obj)
    {

```

```
        service.addPage(obj.getUsername(), obj.getNotes());
        /*
        if(service.checkPage(obj.getUsername(), obj.getNotes())!=null)
        {
            return "Note added successfully!";
        }
        else
        {
            return "Note wasn't added.";
        }
        */
    }
    @PostMapping(path = "/notes")
    public List<String> showNotesController(@RequestBody Stringulescu
username)
    {
        return service.showNotes(username.getUsername());
    }
}
```

Backend Secvență de cod 16 User Controller

```
package com.daspaket.diary.controller;

import com.daspaket.diary.model.User;
import com.daspaket.diary.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping(path = "/user")
public class UserController
{
    @Autowired
    private UserService service;
    @GetMapping(path = "/all")
    public List<User> show()
    {
        //
        return service.getAllUsers();
    }
    @GetMapping(path =("/{id}")
    public User findStudentById(@PathVariable int id)

    {
        return service.findUserById(id);
    }
    @GetMapping(path = "/search")
    public String findStudentById(@RequestParam String username)
    {
        //@RequestBody implies that I must use all the fields and is used
for POST METHOD(add data)
        //example: localhost:8080/user/search?username=Stefan32
        System.out.println(username);
        if(service.findUserByUsername(username) != null)
        {
            return service.findUserByUsername(username).toString();
        }
    }
}
```

```
    }
    else
    {
        return "You request has failed, check your submitted data
again!";
    }
}
@PostMapping(path = "/login")
public String loginController(@RequestBody User user)
{
    if(service.loginProcess(user.getUsername(), user.getPassword()) !=
null)
    {
        //return service.loginProcess(user.getUsername(),
user.getPassword());
        return "Login successful!";
    }
    else
    {
        return "Login failed, please try again or reset your password.";
    }
}
@PutMapping(path = "/add")
public String addUserController(@RequestBody User user)
{
    //se impune si o verificare dupa introducere in DB ptr a fi siguri
ca s-a introdus inregistrarea
    service.addUser(user);
    if(service.findUserByUserNew(user) != null)
    {
        return "New field added with success";
    }
    else
    {
        return "New field failed to be added";
    }
}
@GetMapping(path = "/type")
public String displayTypeController()
{
    String s = "";
    List<String> list = service.displayType();

    s += "[" + list.get(0) + "\", ";
    for(int i = 1; i < list.size() - 1; i++)
    {
        s += "\"";
        s += list.get(i);
        s += "\", ";
    }
    s += "\" + list.get(list.size() - 1) + \"]";
    return s;
    //return service.displayType().toString();
    //return "Types fetched successfully!";
}
}
```

Backend Secvență de cod 17 Question Controller

```
package com.daspaket.diary.controller;

import com.daspaket.diary.model.SecurityQuestion;
import com.daspaket.diary.service.SecurityQuestionService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping(path = "/security")
public class SecurityQuestionController
{
    //
    @Autowired
    SecurityQuestionService service;
    @GetMapping(path = "/all-questions")
    public List<SecurityQuestion> showAll()
    {
        return service.getAllQuestions();
    }
}
```

În al cincilea rând, pentru ca funcționalitatea să fie respectată, s-a introdus și un CORS protocol care asigură că metodele de transmisie a datelor GET, POST, PUT, dar și celelalte, au încuviințarea să fie folosite în contextul aplicație construite.

Backend Secvență de cod 18 CORS

```
package com.daspaket.diary.CORS;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class CORS_protocol implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
            .allowedOrigins("*") // Allow requests from any origin
            .allowedMethods("GET", "POST", "PUT", "DELETE") // Allowed
            HTTP methods
            .allowedHeaders("*"); // Allowed headers
    }
}
```

În final, se va apela programul main care centralizează toate secțiunile de cod enumerate, le compilează și apoi le rulează dacă nu apar probleme de sintaxă sau de funcționalitate.

```
package com.daspaket.diary;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class DiaryApplication {  
  
    public static void main(String[] args)  
    {  
        SpringApplication.run(DiaryApplication.class, args);  
    }  
  
}
```

La finalul acestei secțiuni se impune mențiunea că toate mascările, construite prin metoda Java de încapsulare, sunt dezirabile, dar nu sunt obligatorii. Dacă s-ar face o minimizare a întregului program, s-ar putea observa că toate instanțierile ar putea fi evitate, lăsând doar partea de Repository și partea de Controlere. S-a evitat înadins această procedură întocmai pentru a face tot programul transparent, însemnând ușor de controlat și îmbunătățit, dar și creând un context favorabil pentru debugging.

1.5. POSTMAN

Introducem această secțiune pentru a demonstra independent de frontend, parte care va fi tratată la urmă, așa cum impune și ierarhia firească enunțată la început, funcționalitatea protocolului de comunicare HTTP. POSTMAN este o aplicație care facilitează comunicația dintre server și programul scris în IntelliJ, acesta este capabil să trimită cereri într-un mod facil către server. Deși acest soft a fost folosit, funcționalitatea programului dezvoltat așa cum reiese până în acest moment, nu depinde de acest soft, dar este un tool cu care vom demonstra funcționalitatea programului.

Așadar, se va enunța interogarea în MySQL, precum și rezultatul ei, iar ulterior se va înfăptui același demers prin intermediul POSTMAN.

Exemplul 1:

Se vor afișa notițele scrise de utilizatorul Ștefan32. Notițele sunt parte a tabelului Notes, iar utilizatorul Ștefan se găsește în tabelul User. Cele 2 tabele sunt corelate cu ajutorul unei chei străine.

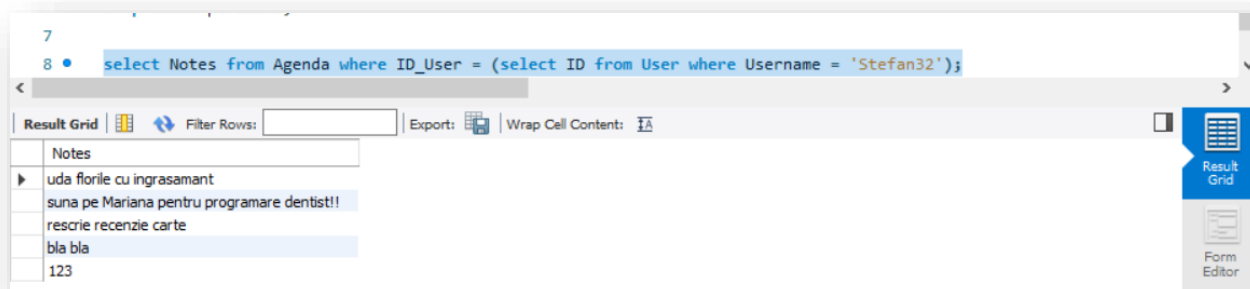


Fig 1.5.1
Query în MySQL

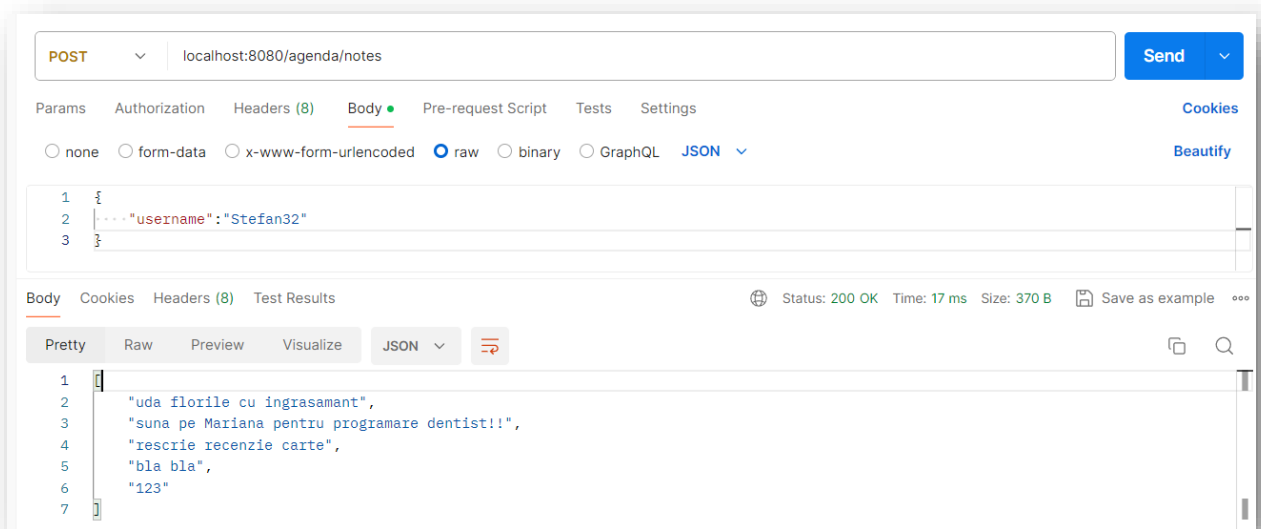


Fig 1.5.2
POSTMAT cu request POST

Exemplul 2:

Se va adăuga un utilizator cu datele din figura de mai jos.

- username: Marian Godina
- password: regele-asfaltului
- email: zarnestii_mari@yahoo.ro
- type: Premium

Se observă că nu se adaugă un utilizator după toate câmpurile din tabel, de exemplu, ID-ul se incrementează singur fără a fi nevoie de a fi adăugat împreună cu celelalte câmpuri.

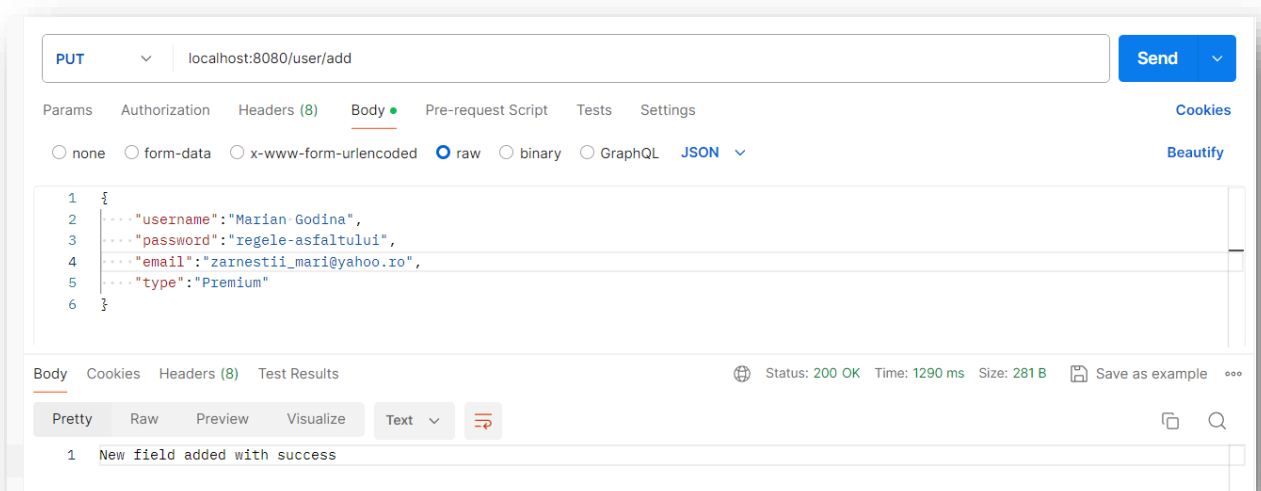
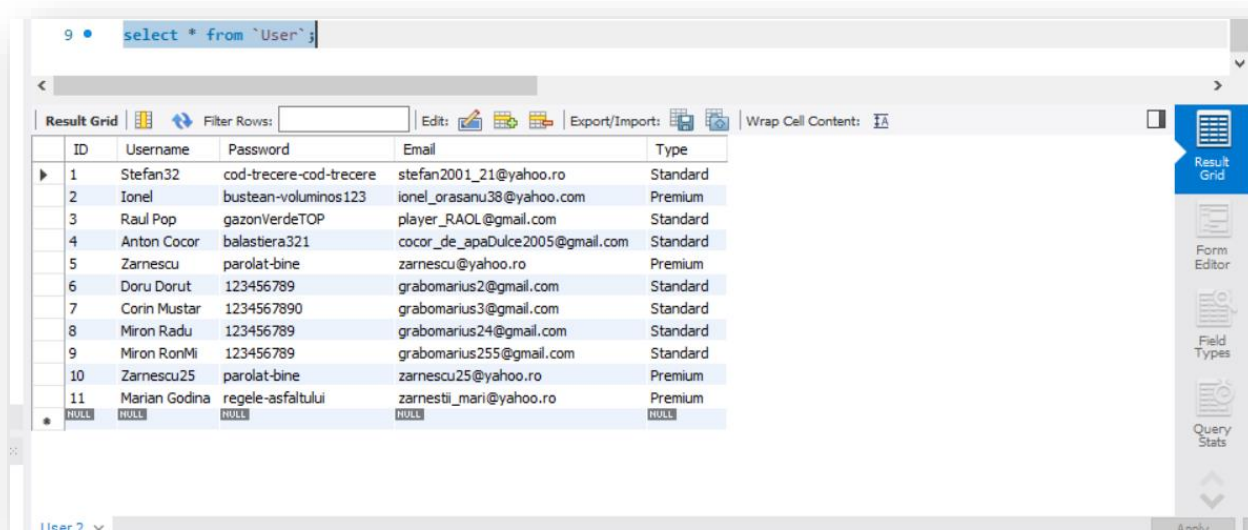


Fig 1.5.3
POSTMAT cu request PUT



ID	Username	Password	Email	Type
1	Stefan32	cod-trecere-cod-trecere	stefan2001_21@yahoo.ro	Standard
2	Ionel	bustean-voluminos123	ionel_orasanu38@yahoo.com	Premium
3	Raul Pop	gazonVerdeTOP	player_RAOL@gmail.com	Standard
4	Anton Cocor	balastiera321	cocor_de_apaDulce2005@gmail.com	Standard
5	Zarnescu	parolat-bine	zarnescu@yahoo.ro	Premium
6	Doru Dorut	123456789	grabomarius2@gmail.com	Standard
7	Corin Mustar	1234567890	grabomarius3@gmail.com	Standard
8	Miron Radu	123456789	grabomarius24@gmail.com	Standard
9	Miron RonMi	123456789	grabomarius255@gmail.com	Standard
10	Zarnescu25	parolat-bine	zarnescu25@yahoo.ro	Premium
11	Marian Godina	regele-asfaltului	zarnestii_mari@yahoo.ro	Premium

Fig 1.5.4
MySQL pentru a verifica adăugarea

În concluzie, POSTMAN manevrează request-urile HTTP și facilitează comunicația server – program backend. O altă mențiune este că pentru a trimite date către server s-a folosit metoda text raw după formă JSON, javascript object notation.

1.6. Frontend

Partea finală și cea mai anevoie de programat este partea de interfață, denumită și API, care permite utilizatorului să altereze direct baza de date și, în general, să se deservească de aceasta pentru a scrie notițe în agendă, având și posibilitatea să își creeze un cont gratuit în cazul în care nu are și dorește să aibă parte de o agendă virtuală. Se vor scrie în cele ce urmează doar fișierele cu care s-a lucrat în mod direct, iar cele generate de Maven, care se ocupă de dependențele necesare rulării programului, dar și cele generate de Node.js, script-ul care face un bundle generic cu care programatorul lucrează. Pentru editare textului s-a folosit Microsoft Visual Code. Ordinea în care vor fi prezentate secțiunile de cod este ordinea în care acestea sunt plasate în foldere de sus în jos.

Framework-ul React a fost folosit pentru compilarea codului de frontend. O alternativă pentru React ar putea fi Angular, însă lucrul programatorului Java este mai dificil, deși TypeScript, supersetul Javascript, face lucrul cu obiecte mai facil. O mențiune importantă este că atunci când programul este încărcat în browser, acesta este convertit implicit în Javascript, așadar deși lucrul programatorului este cu TypeScript, browser-ul nu are posibilitatea de a-l înțelege fiind o îmbunătățire pentru Javascript relativ recentă, iar browser-ele încă nu dețin capacitatea de-al percepe ca atare.

O librărie, și totodată un tool convenabil, este axios care permite accesul la hyperlink-urile de tipul GET, POST, PUT foarte ușor, având procedura de lucru cu acestea predefinită.

Secvență de cod 1 Navbar

```
import React from 'react'
import { Link } from 'react-router-dom'

export default function Navbar({ onLogout }) {
  const handleLogout = () => {
    // Call the onLogout function passed from the parent component
    onLogout();
  };
  return (
    <div>
      <nav className="navbar navbar-expand-lg navbar-dark bg-primary">
        <div className="container-fluid">
          <Link className='btn btn-outline-light' onClick={handleLogout}
to="/">Log out</Link>
        </div>
      </nav>
    </div>
  )
}
```

Secvență de cod 2 Home

```
import React, { useEffect, useState } from 'react'
import axios from 'axios';

export default function Home() {
  return (
    <div className='container'>
    </div>
  )
}
```

Secvență de cod 3 Display Types

```
import React, { useState, useEffect } from 'react';

function RadioButtons() {
  const [types, setTypes] = useState([]);
  const [selectedType, setSelectedType] = useState('');

  useEffect(() => {
    // Fetch data from localhost:8080/user/type
    fetch('http://localhost:8080/user/type')
      .then(response => response.json())
      .then(data => {
        // Assuming the response data is an array of strings
        representing types
        setTypes(data);
        // Set the default selected type
        if (data.length > 0) {
          setSelectedType(data[0]);
        }
      })
  })
}
```

```
        .catch(error => console.error('Error fetching types:', error));
    }, []);

    const handleTypeChange = (e) => {
        setSelectedType(e.target.value);
    };

    return (
        <div>
            <label>User Type:</label><br />
            {types.map((type, index) => (
                <div key={index}>
                    <input
                        type="radio"
                        id={type}
                        name="type"
                        value={type}
                        checked={selectedType === type}
                        onChange={handleTypeChange}
                    />
                    <label htmlFor={type}>{type}</label><br />
                </div>
            ))}
        </div>
    );
}

export default RadioButtons;
```

Secvență de cod 4 Login

```
import React, { useState } from 'react';
import axios from 'axios';

export default function Login({ onLogin }) {
    const [username, setUsername] = useState('');
    const [password, setPassword] = useState('');
    const [responseData, setResponseData] = useState(null);
    const [error, setError] = useState(null);

    const handleSubmit = async (event) => {
        event.preventDefault(); // Prevent default form submission behavior

        try {
            // Send data to the server
            const response = await
            axios.post('http://localhost:8080/user/login', {
                username: username,
                password: password
            });

            // Handle the response
            onLogin(username, response.data);
            setResponseData(response.data);
            setError(null);
            setUsername('');
            setPassword('');

        } catch (error) {
```

```
        // Handle errors
        onLogin(null, null);
        console.error('Error:', error);
        setResponseData(null);
        setError('An error occurred while sending data to the server.');
```

```
    }
  };

  return (
    <div className="d-flex justify-content-center align-items-center"
      style={{ minHeight: "10vh" }}>
      <div>
        <h2>Login</h2>
        <div style={{ maxWidth: "300px" }}>
          <form onSubmit={handleSubmit} id='login-form'>
            <div className="mb-2">
              <label className="ml-4 pr-2">
                Username:
                <input type="text" className="form-control"
value={username} onChange={(e) => { setUsername(e.target.value); }} />
              </label>
            </div>
            <div className="mb-2">
              <label className="ml-4 pr-2">
                Password:
                <input type="password" className="form-
control" value={password} onChange={(e) => { setPassword(e.target.value); }} />
              </label>
            </div>
            <button type="submit" className='mt-2 btn btn-
primary'>Login</button>
          </form>
          {responseData && (
            <div>
              <h3>Response from Server:</h3>
              <pre>{JSON.stringify(responseData, null,
2)}</pre>
            </div>
          )}
          {error && <div className="ml-4 text-danger">Error:
{error}</div>}
        </div>
      </div>
    </div>
  );
}
```

Secvență de cod 5 Preproces

```
import React from 'react';

const buttonStyle = {
  cursor: 'pointer',
  backgroundColor: '#53b5ff',
  color: 'white',

  transition: 'background-color 0.3s ease',
  border: '2px solid #007bff', // Added border with color
```

```
padding: '0.5rem 1rem', // Adjusted padding for a smaller button
borderRadius: '0.25rem', // Adjusted border radius
};

const buttonHoverStyle = {
  backgroundColor: '#0056b3',
};

const Heading = {
  fontSize: '1.3rem',
};

export default function Preprocess({ handleSignUp, handleLogin }) {
  return (
    <div className="d-flex justify-content-center align-items-center"
    style={{ minHeight: "5vh" }}>
      <div>
        <h1 style={Heading}>Choose method of authentication:</h1>
        <div style={{ ...buttonStyle }} onClick={handleSignUp}>Sign Up</div>
        <div style={{ ...buttonStyle }} onClick={handleLogin}>Log In</div>
      </div>
    </div>
  );
}
```

Secvență de cod 6 Sign Up

```
import React, { useState } from 'react';
import axios from 'axios';
import DisplayTypes from './DisplayTypes';

export default function Signup({ onSignup }) {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [email, setEmail] = useState('');
  const [type, setType] = useState('Standard');
  const [responseData, setResponseData] = useState(null);
  const [error, setError] = useState(null);

  const handleSubmit = async (event) => {
    event.preventDefault(); // Prevent default form submission behavior

    try {
      // Send data to the server
      const response = await
    axios.put('http://localhost:8080/user/add', {
      username: username,
      password: password,
      email: email,
      type: type
    });

    // Handle the response
    onSignup(username, response.data);
    setResponseData(response.data);
    setError(null);
    setUsername(''); setPassword(''); setEmail(''); setType('');
  }
}
```

```
    } catch (error) {
      // Handle errors
      onSignup(null, null);
      console.error('Error:', error);
      setResponseData(null);
      setError('An error occurred while sending data to the server.');
```

```
    };

    return (
      <div className="d-flex justify-content-center align-items-center"
style={{ minHeight: "10vh" }}>
        <div className="ml-4">
          <h2>Signup</h2>
          <div style={{ maxWidth: "300px" }}>
            <form onSubmit={handleSubmit}>
              <div className="mb-2">
                <label className='ml-4 pr-2'>
                  Email:
                  <input type="email" className="form-control"
value={email} onChange={(e) => setEmail(e.target.value)} />
                </label>
              </div>
              <div className="mb-2">
                <label className='ml-4 pr-2'>
                  Username:
                  <input type="text" className="form-control"
value={username} onChange={(e) => setUsername(e.target.value)} />
                </label>
              </div>
              <div className="mb-2">
                <label className='ml-4 pr-2'>
                  Password:
                  <input type="password" className="form-
control" value={password} onChange={(e) => setPassword(e.target.value)} />
                </label>
              </div>
              <DisplayTypes /><br />
              <button type="submit" className='btn btn-
primary'>Sign Up</button>
            </form>
            {responseData && (
              <div>
                <h3>Response from Server:</h3>
                <pre>{JSON.stringify(responseData, null,
2)}</pre>
              </div>
            )}
            {error && <div className="text-danger">Error:
{error}</div>}
          </div>
        </div>
      </div>
    );
  }
}
```

Secvență de cod 7 Diary.css

```
#listCSS {  
  overflow-y: auto;  
  /* Enable vertical scrollbar */  
  max-height: 100px;  
  /* Set maximum height to limit expansion */  
}
```

Secvență de cod 8 Diary.js

```
import React, { useEffect, useState } from 'react';  
import axios from 'axios';  
import Note from './Note';  
  
export default function Diary({ usernameDiary }) {  
  
  const [allNotes, setAllNotes] = useState([]);  
  // State to store all notes  
  const fetchNotes = async () => {  
    try {  
      const response =  
        await axios.post('http://localhost:8080/agenda/notes', {  
          username: usernameDiary  
        });  
      if (Array.isArray(response.data)) {  
        setAllNotes(response.data);  
        // Update allNotes state with the received data  
      } else {  
        console.error('Invalid data received:', response.data);  
        setError('Invalid data received from the server.');      }  
    } catch (error) {  
      console.error('Error fetching notes:', error);  
      setError('An error occurred while fetching notes.');    }  
  };  
  
  useEffect(() => {  
    // Update datetime la fiecare secunda  
    const updateDateTime = () => {  
      const now = new Date();  
      const datetimeElement = document.getElementById('datetime');  
      if (datetimeElement) {  
        datetimeElement.textContent = now.toLocaleString();  
      }  
    };  
  
    updateDateTime(); // Initial call  
    const intervalId = setInterval(updateDateTime, 1000);  
    // Update la fiecare secunda  
  
    return () => clearInterval(intervalId);  
  }, []);  
  
  const [username, setUsername] = useState('');  
  const [notes, setNotes] = useState('');  
  const [error, setError] = useState('');  
  const [view, setView] = useState(false);  
  
  const handleSubmit = async (event) => {  
    //submit paseaza datele pentru interogare(in cazul asta adaugare de field)  
    event.preventDefault(); // Prevent default form submission behavior
```



```
try {
  // Send data to the server
  const response =
    await axios.put('http://localhost:8080/agenda/add-note', {
      username: usernameDiary,
      notes: notes
    });

  // Handle the response
  // setUsername(usernameDiary);
  setNotes("note added succesfully!!");
  fetchNotes();
  // Reload the page after notes are saved
  // window.location.reload();
} catch (error) {
  // Handle errors
  console.error('Error:', error);
  setError('An error occurred while sending data to the server. ');
  // Reload the page after notes are saved
  // window.location.reload();
}
};

const btnClick = () => {
  if (view) {
    setView(false);
    console.log('Dont View user notes');
  }
  else {
    setView(true);
    console.log('View user notes');
  }
};

useEffect(() => {
  fetchNotes();
}, [usernameDiary]);

return (
  <>
    <link
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
      rel="stylesheet"
    />
    <div className="user-info">
      <span>User: {usernameDiary}</span>
      <br />
      <span id="datetime"></span>
    </div>
    <div className="container">
      <div className="row">
        <div className="col-md-8 offset-md-2">
          <form onSubmit={handleSubmit}>
            <textarea
              className="form-control"
              id="notepad"
              rows="10"
              value={notes}
              onChange={e => setNotes(e.target.value)}
            ></textarea>
            <button className='mt-2 mb-3 btn btn-primary' type='submit'>Save
notes</button>
          </form>
          <div className="scrollable-notes">
            <button onClick={btnClick} className='btn btn-primary mb-3'>View all
notes</button>
          </div>
        </div>
      </div>
    </div>
  </>
);
```

```
        <ul>
          {allNotes.map((note, index) => (
            <div key={index} className="note" style={{ display: (view) ? 'block' :
'none' }}>
              <div id="listCSS">
                <li>
                  {view && <Note note={note} />} {/* Conditional rendering of Note
component */}
                </li>
              </div>
            </div>
          ))}
        </ul>
      </div>
    </div>
  </div>
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  </>
  );
}
```

Secvență de cod 9 Note.js

```
import React from 'react';

const Note = ({ note }) => {
  return (
    <div className="note-container"
      style={{ maxHeight: '50px', overflowY: 'auto'}}>
      {note}
    </div>
  );
};

export default Note;
```

Secvență de cod 10 App.js

```
import './App.css';
import './node_modules/bootstrap/dist/css/bootstrap.min.css';
import Navbar from './layout/Navbar';
import Home from './pages/Home';
import { Navigate, BrowserRouter as Router, Routes, Route } from 'react-
router-dom';

import Preprocess from './process/Preprocess';
import Signup from './process/Signup';
import Login from './process/Login';
import Diary from './users/Diary';
import React, { useState } from 'react';
import './App.css';

export default function App() {
  const [signup, setSignup] = useState(false);
  const [login, setLogin] = useState(false);
```

```
const [route_signup, setRouteSignUp] = useState(false);
const [route_login, setRouteLogIn] = useState(false);
const [usernameDiary, setUsernameDiary] = useState(''); // State to store
username for Diary
//const navigate = useNavigate(); // Use useHistory hook

const handleSignUp = () => {
  // Logic for handling sign-up action
  setSignup(true);
  setLogin(false);
  console.log('Sign Up');
};

const handleLogin = () => {
  // Logic for handling login action
  setSignup(false);
  setLogin(true);
  console.log('Log In');
};

const handleSignup1 = (username, message) => {
  // Do something with the updated values
  console.log('sign up username:', username);
  console.log('sign up message:', message);
  if (message === "New field added with success") {
    setRouteSignUp(true);
    //navigate('/diary');
    setUsernameDiary(username);
  }
  else {
    setRouteSignUp(false);
    setUsernameDiary("default");
  }
};

const handleLogin1 = (username, message) => {
  // Do something with the updated values
  console.log('log in username:', username);
  console.log('log in message:', message);
  if (message === "Login successful!") {
    setRouteLogIn(true);
    //navigate('/diary');
    setUsernameDiary(username);
  }
  else {
    setRouteLogIn(false);
    setUsernameDiary("default");
  }
};

const handleLogout = () => {
  // Reset all state values to their initial state
  setSignup(false);
  setLogin(false);
  setRouteSignUp(false);
  setRouteLogIn(false);
  setUsernameDiary('');
};
```

```
return (
  <div className="App">
    <Router>
      <Navbar />
      <Routes>
        <Route exact path="/" element={<Home/>}></Route>
        <Route exact path="/diary" element={<Diary
usernameDiary={usernameDiary} />}></Route>
      </Routes>
      <div id='auth-area' style={{ display: (route_login || route_signup)
? 'none' : 'block' }}>
        <Preprocess handleSignUp={handleSignUp} handleLogin={handleLogin}
/>
        {signup && <Signup onSignup={handleSignup1} />}
        {login && <Login onLogin={handleLogin1} />}
      </div>
      <div>
        {(route_login || route_signup) && <Navigate to="/diary" />}
      </div>
    </Router>
  </div>
);
}
```

Secvență de cod 11 manifest.json

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "logo192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "logo512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

1.7. Manual de utilizare

Utilizatorul va accesa aplicația prin intermediul unei ferestre de logare. Acesta se poate autentifica sau crea un cont nou. Crearea unui cont va fi demonstrată printr-un request cu ajutorul POSTMAN.

Choose method of authentication:

Sign Up

Log In

Signup

Email:
test-test20249@gmail.ro

Username:
Corin Olteanul

Password:
.....

User Type:
☒ Standard
☐ Premium

Sign Up

Fig 1.7.1
Creare cont

Consecința firească este adăugarea în baza de date a acestui după cum urmează în figura 1.7.2.

ID	Username	Password	Email	Type
1	Stefan32	cod-trecere-cod-trecere	stefan2001_21@yahoo.ro	Standard
2	Ionel	bustean-voluminos123	ionel_orasanu38@yahoo.com	Premium
3	Raul Pop	gazonVerdeTOP	player_RAOL@gmail.com	Standard
4	Anton Cocor	balastiera321	cocor_de_apaDulce2005@gmail.com	Standard
5	Zarnescu	parolat-bine	zarnescu@yahoo.ro	Premium
6	Doru Dorut	123456789	grabomarius2@gmail.com	Standard
7	Corin Mustar	1234567890	grabomarius3@gmail.com	Standard
8	Miron Radu	123456789	grabomarius24@gmail.com	Standard
9	Miron RonMi	123456789	grabomarius255@gmail.com	Standard
10	Zarnescu25	parolat-bine	zarnescu25@yahoo.ro	Premium
11	Marian Godina	regele-asfaltului	zarnesti_mari@yahoo.ro	Premium
12	rgr45gh5th...	h56th56b56h56h56h56...	grabomarius2rrrr@gmail.com	Standard
13	g54g45gdve	445gsadcascescecec	f34g34@yahoo.ro	Standard
14	Corin Olteanul	123456789	test-test20249@gmail.ro	Standard

Fig 1.7.2
Schimbare în baza de date în User

Figura 1.7.3 arată trecerea la un textbox în care utilizatorul își poate scrie notițele și poate vedea ceasul actual și numele său de utilizator. Următoarele figuri vor arăta celelalte funcționalități.

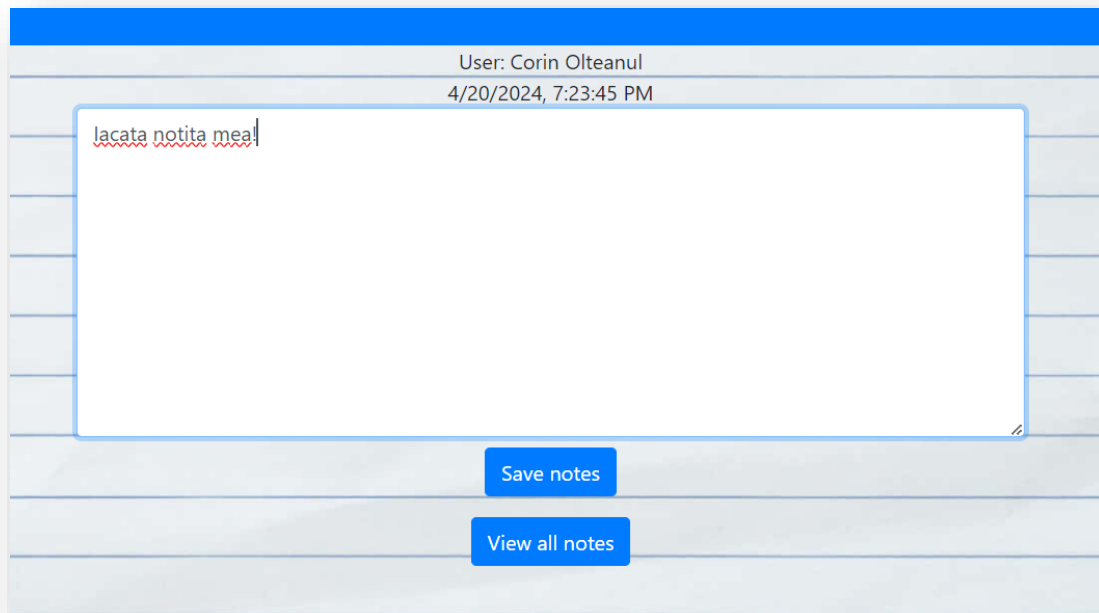


Fig 1.7.3
Zona de scriere a notițelor

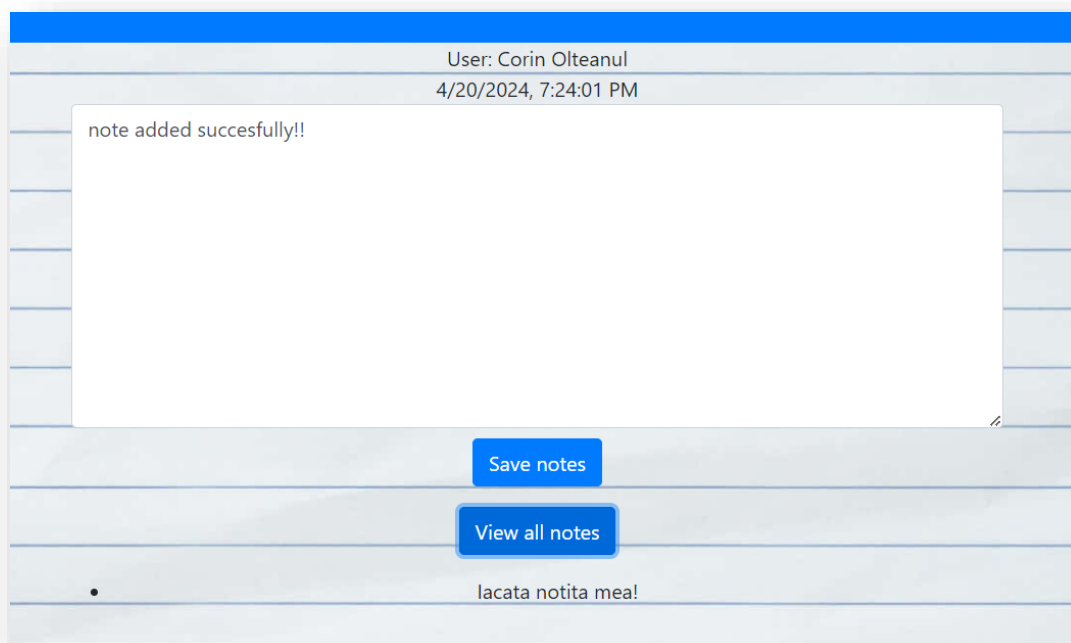


Fig 1.7.4
Mesaj de succes și memorare în agendă

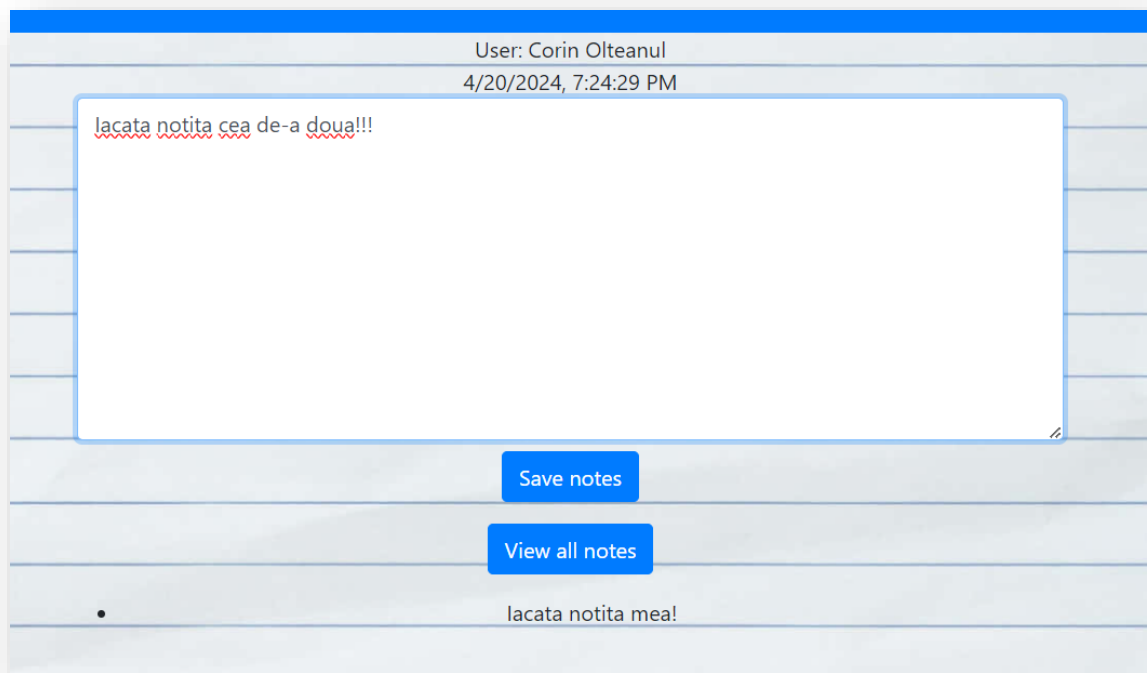


Fig 1.7.5
Repetare aceluiași demers

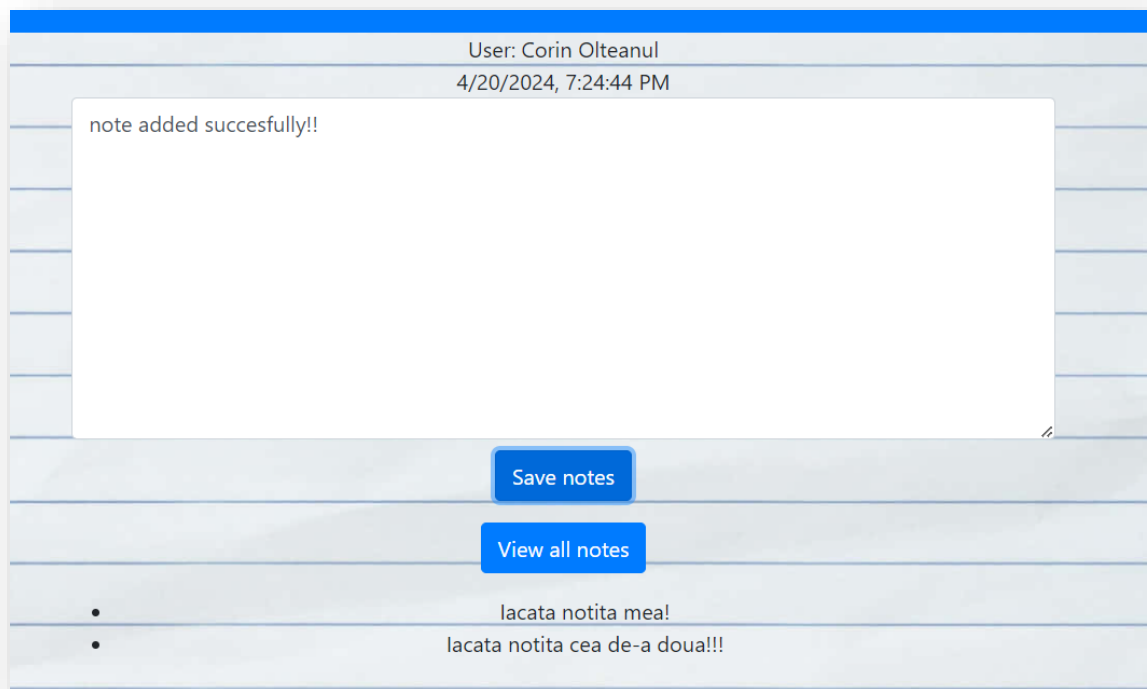


Fig 1.7.6

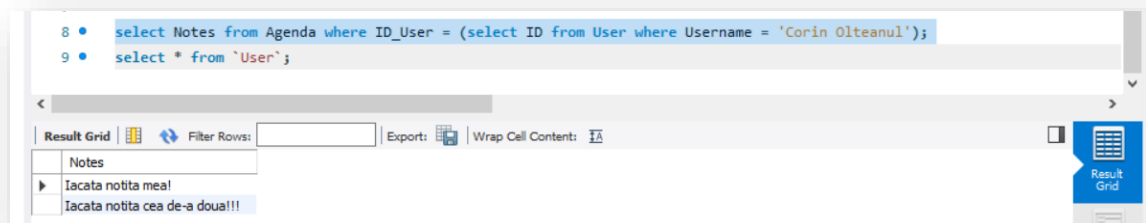


Fig 1.7.7
Confirmarea salvării notițelor

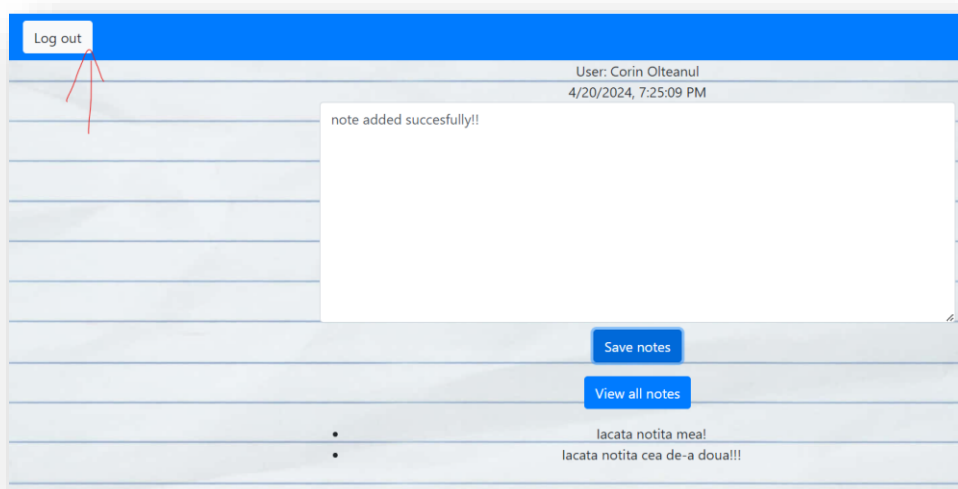
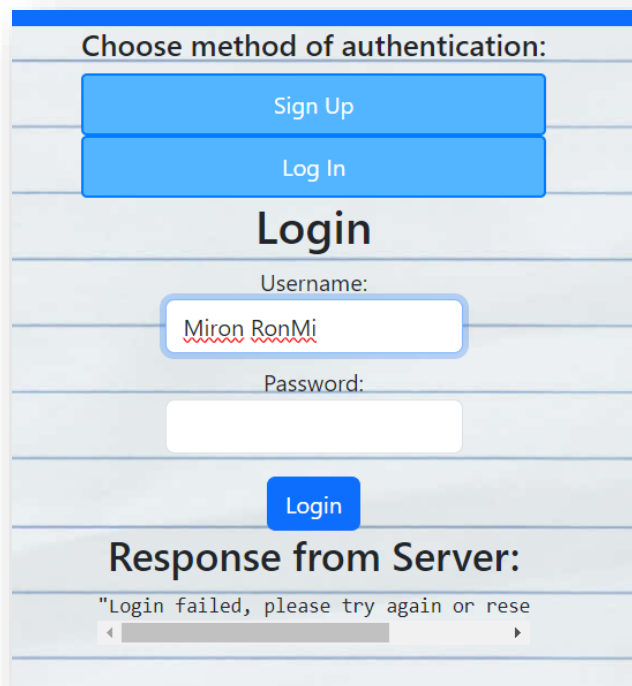


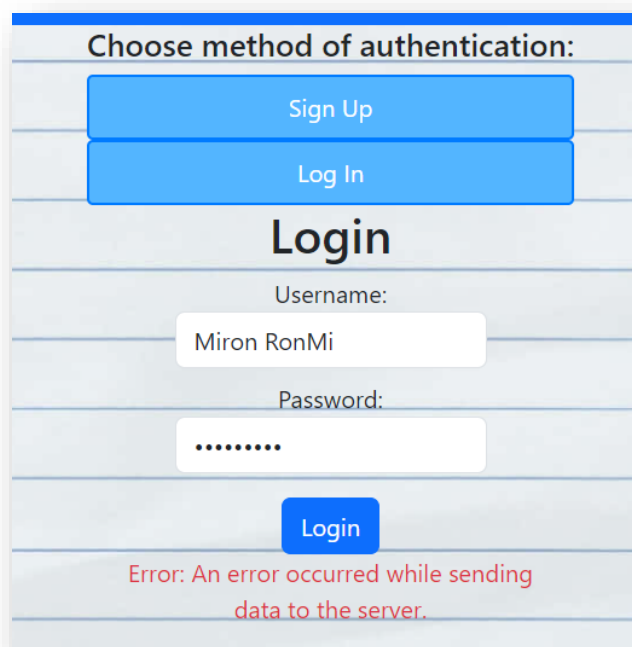
Fig 1.7.8
Delogare/ieșire din cont

În continuare se va face autentificarea pe baza unui cont deja existent și se vor proba mesajele de eroare ce pot surveni în cazul în care datele nu au fost introduse corect ori dacă serverul este inactiv or inaccesibil din diverse motive, însemnând că requestul HTTP nu va avea destinatar, astfel utilizatorul va fi întâmpinat cu un mesaj de eroare de comunicare cu serverul, deși poate datele introduse sunt corecte și conforme cu realitatea, însă nu se pot verifica.



The screenshot shows a web interface for authentication. At the top, there's a section titled "Choose method of authentication:" with two blue buttons: "Sign Up" and "Log In". Below this is the "Login" section. It has a "Username:" label followed by a text input field containing "Miron RonMi". Below the username field is a "Password:" label followed by a password input field. A blue "Login" button is positioned below the password field. At the bottom, there's a section titled "Response from Server:" which displays a message in a scrollable area: "Login failed, please try again or rese".

Fig 1.7.9
Date neconforme sau incorecte



The screenshot shows the same web interface as Fig 1.7.9. The "Username:" field contains "Miron RonMi" and the "Password:" field is filled with dots. The blue "Login" button is visible. Below the login button, there is a red error message: "Error: An error occurred while sending data to the server."

Fig 1.7.10
Eroare de comunicare cu serverul