Object Oriented Analysis and Design using Java

(UE20CS352)

Mini Project

Project Title: Blogging Application
Section: K

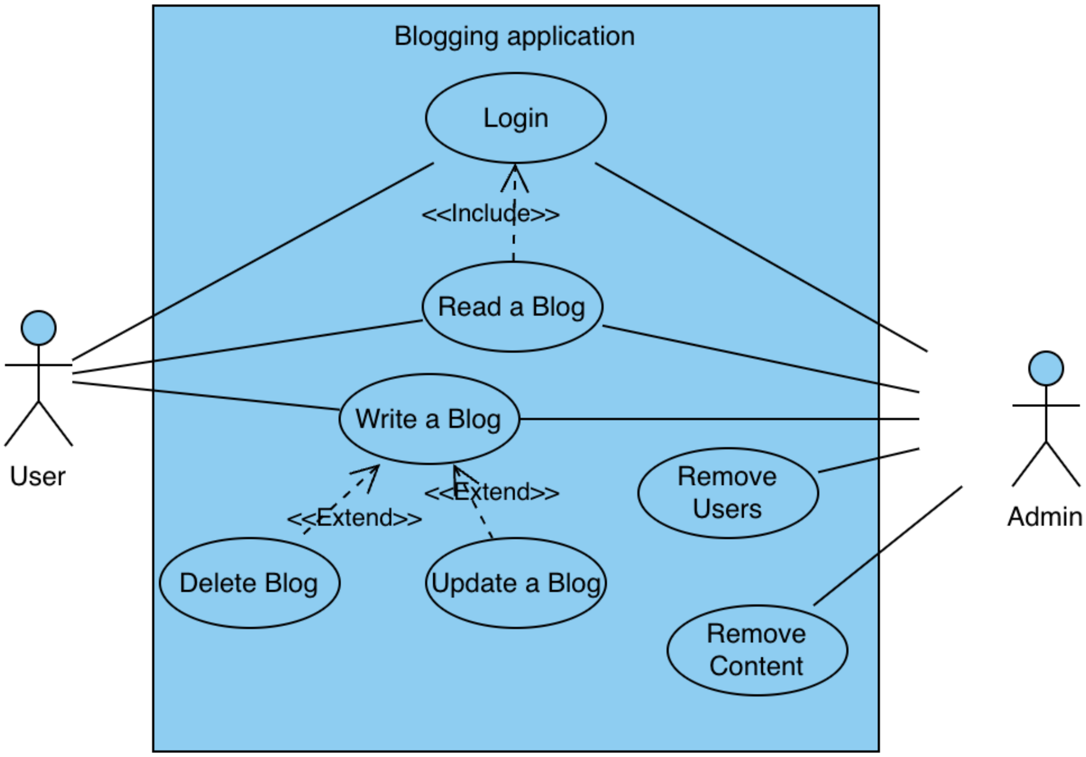
Team Members:

Name	**SRN**
Adithya M	PES1UG20CS621
Achyuta Nandakumar Katta	PES1UG20CS618
B G Sourav	PES1UG20CS629

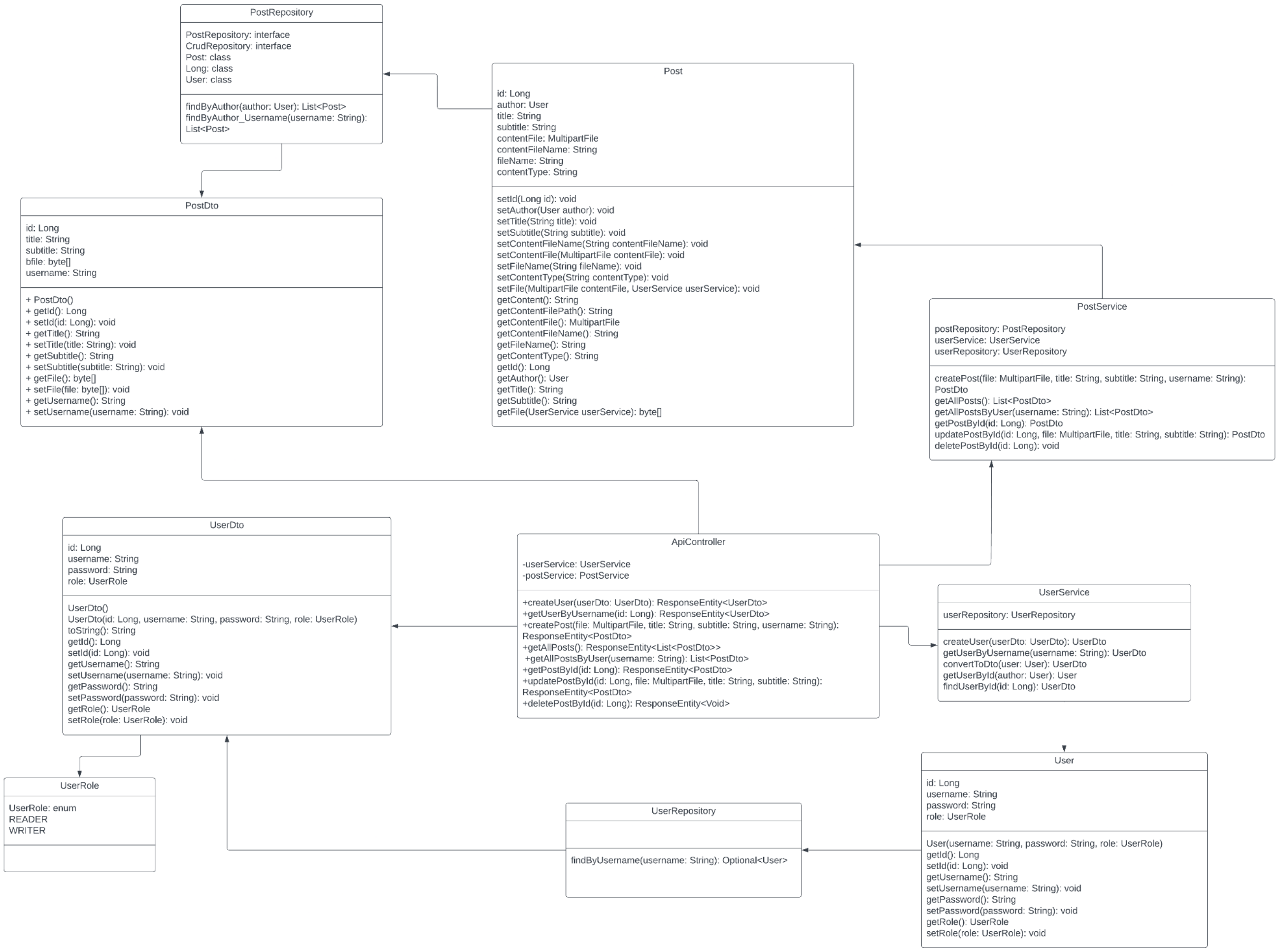
Synopsis

This project involves the development of a Java-based blogging application using the Model-View-Controller (MVC) architecture and design patterns to ensure a robust and maintainable codebase. The application offers a range of features to users, including the ability to create an account, login/signup, post blogs with an optional file attachment, and modify their own blogs. The blog data is stored efficiently in a PostgreSQL database on the backend. Users can also view other users' blogs. The application ensures the security of user data by implementing authentication and authorization mechanisms. Extensive use of design patterns, such as the Factory, Singleton, and Observer patterns, has been made in the application code to ensure a modular and scalable codebase. The use of the MVC architecture and design patterns ensures that the application is efficient, maintainable, and user-friendly. Overall, this project offers a feature-rich and well-designed blogging application that meets the needs of users while adhering to best practices in Java development.

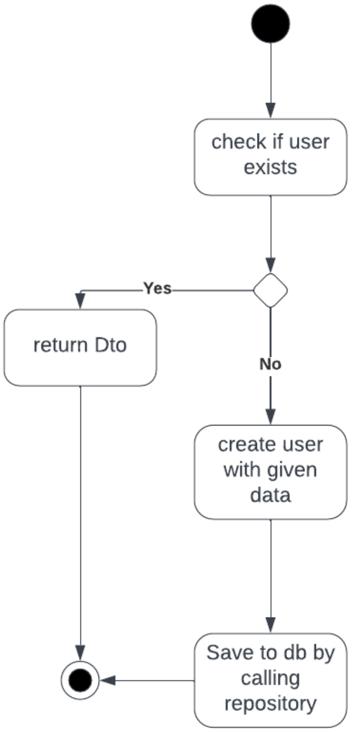
USE CASE DIAGRAM

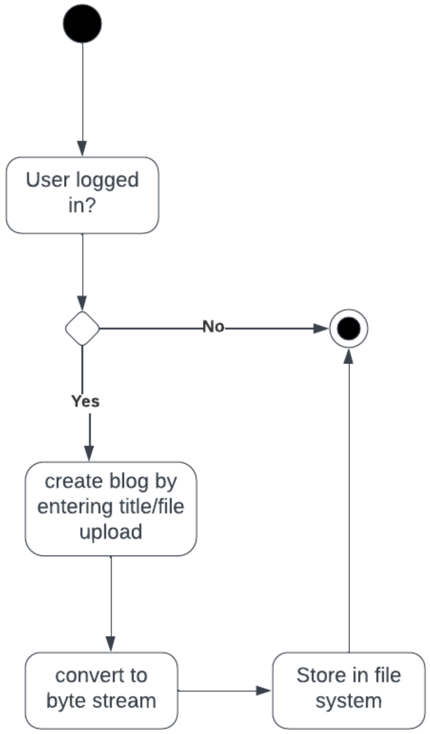


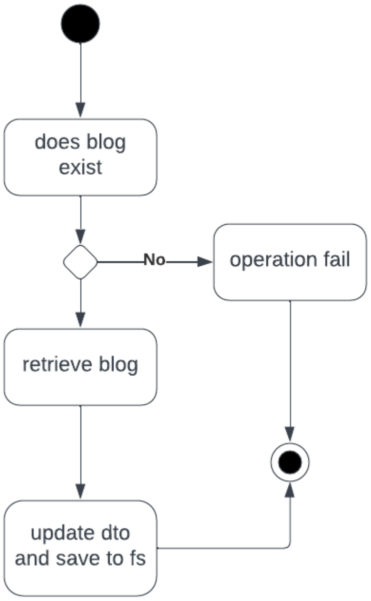
CLASS DIAGRAM


PostRepository
PostRepository: interface
CrudRepository: interface
Post: class
Long: class
User: class
findByAuthor(author: User): List<Post>
findByAuthor_Username(username: String):
List<Post>
Post
id: Long
author: User
title: String
subtitle: String
contentFile: MultipartFile
contentFileName: String
fileName: String
contentType: String
setId(Long id): void
setAuthor(User author): void
setTitle(String title): void
setSubtitle(String subtitle): void
setContentFileName(String contentFileName): void
setContentFile(MultipartFile contentFile): void
setFileName(String fileName): void
setContentType(String contentType): void
setFile(MultipartFile contentFile, UserService userService): void
getContent(): String
getContentFilePath(): String
getContentFile(): MultipartFile
getContentFileName(): String
getFileName(): String
getContentType(): String
getId(): Long
getAuthor(): User
getTitle(): String
getSubtitle(): String
getFile(UserService userService): byte[]
PostDto
id: Long
title: String
subtitle: String
bfile: byte[]
username: String
+ PostDto()
+ getId(): Long
+ setId(id: Long): void
+ getTitle(): String
+ setTitle(title: String): void
+ getSubtitle(): String
+ setSubtitle(subtitle: String): void
+ getFile(): byte[]
+ setFile(file: byte[]): void
+ getUsername(): String
+ setUsername(username: String): void
PostService
postRepository: PostRepository
userService: UserService
userRepository: UserRepository
createPost(file: MultipartFile, title: String, subtitle: String, username: String):
PostDto
getAllPosts(): List<PostDto>
getAllPostsByUser(username: String): List<PostDto>
getPostById(id: Long): PostDto
updatePostById(id: Long, file: MultipartFile, title: String, subtitle: String): PostDto
deletePostById(id: Long): void
UserDto
id: Long
username: String
password: String
role: UserRole
UserDto()
UserDto(id: Long, username: String, password: String, role: UserRole)
toString(): String
getId(): Long
setId(id: Long): void
getUsername(): String
setUsername(username: String): void
getPassword(): String
setPassword(password: String): void
getRole(): UserRole
setRole(role: UserRole): void
ApiController
-userService: UserService
-postService: PostService
+createUser(userDto: UserDto): ResponseEntity<UserDto>
+getUserByUsername(id: Long): ResponseEntity<UserDto>
+createPost(file: MultipartFile, title: String, subtitle: String, username: String):
ResponseEntity<PostDto>
+getAllPosts(): ResponseEntity<List<PostDto>>
+getAllPostsByUser(username: String): List<PostDto>
+getPostById(id: Long): ResponseEntity<PostDto>
+updatePostById(id: Long, file: MultipartFile, title: String, subtitle: String):
ResponseEntity<PostDto>
+deletePostById(id: Long): ResponseEntity<Void>
UserService
userRepository: UserRepository
createUser(userDto: UserDto): UserDto
getUserByUsername(username: String): UserDto
convertToDto(user: User): UserDto
getUserById(author: User): User
findUserById(id: Long): UserDto
UserRole
UserRole: enum
READER
WRITER
UserRepository
findByUsername(username: String): Optional<User>
User
id: Long
username: String
password: String
role: UserRole
User(username: String, password: String, role: UserRole)
getId(): Long
setId(id: Long): void
getUsername(): String
setUsername(username: String): void
getPassword(): String
setPassword(password: String): void
getRole(): UserRole
setRole(role: UserRole): void

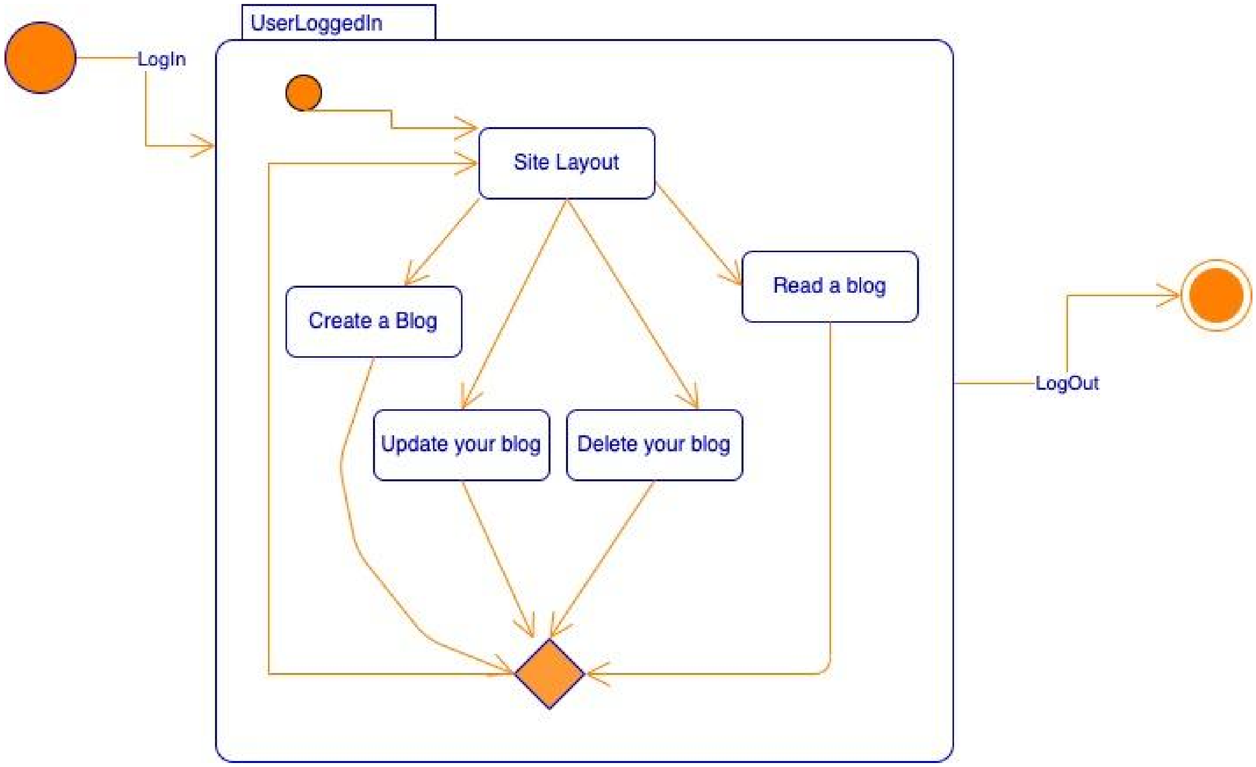
ACTIVITY DIAGRAM







STATE DIAGRAM:



MODEL – VIEW – CONTROLLER ARCHITECTURE:

The Model View Controller (MVC) design pattern specifies that an application consists of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects. MVC is more of an architectural pattern, but not for complete application. MVC mostly relates to the UI / interaction layer of an application. You're still going to need a business logic layer, maybe some service layer and data access layer.

- The **Model** contains only the pure application data, it contains no logic describing how to present the data to a user. (It's just data that is shipped across the application like for example from back-end server view and from front-end view to the database. In java programming, models can be represented by the use of POJO (Plain-old-java-object) which is a simple java class.

- The **View** presents the model’s data to the user. The view knows how to access the model’s data, but it does not know what this data means or what the user can do to manipulate it. View just represents, displays the application’s data on screen. View pages are generally in the format of .html or .jsp in java programming (which is flexible).

- The **Controller** exists between the view and the model. It is where the actual business logic is written. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events. In most cases, the reaction is to call a method on the model. Since the view and the model are connected through a notification mechanism, the result of this action is then automatically reflected in the view.

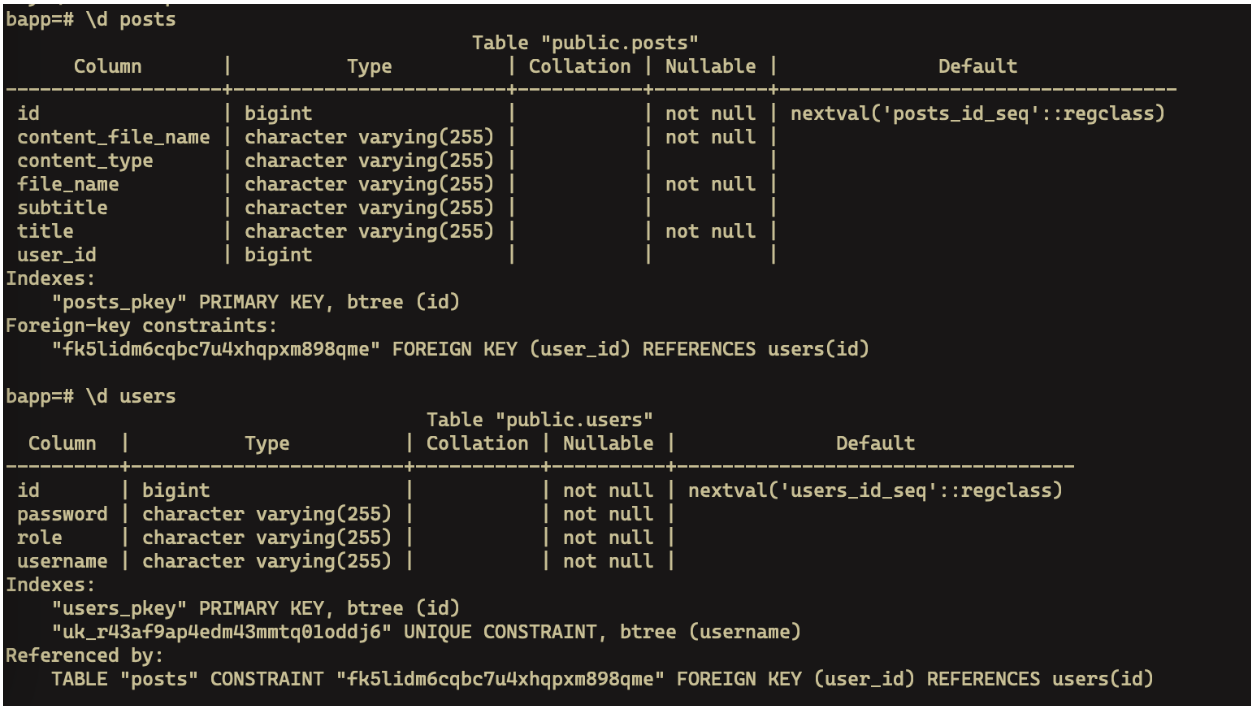
DESIGN PRINCIPLES:

We have utilized the Single Responsibility principle to isolate the working of different components in our backend.
We have utilized Dependency inversion as part of spring JPA interfaces when we're creating an ORM for our database tables

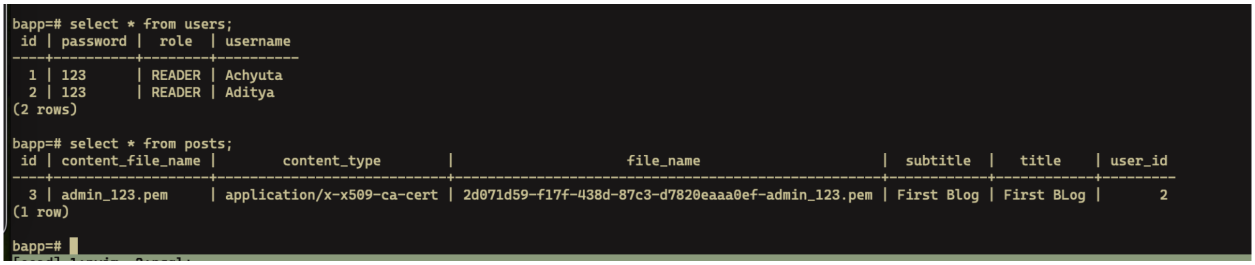
Design Patterns:

Singleton Pattern:We are using a login system. We are making use of the singleton pattern to make sure that only one user instance is created at a time.
Factory Pattern: The project utilizes the Factory design pattern to create blog objects (PostDto and UserDto) in a flexible and modular way.
Facade Pattern:The project utilizes the Facade design pattern to encapsulate the implementation details of the blog object and provide a simplified interface to the users.

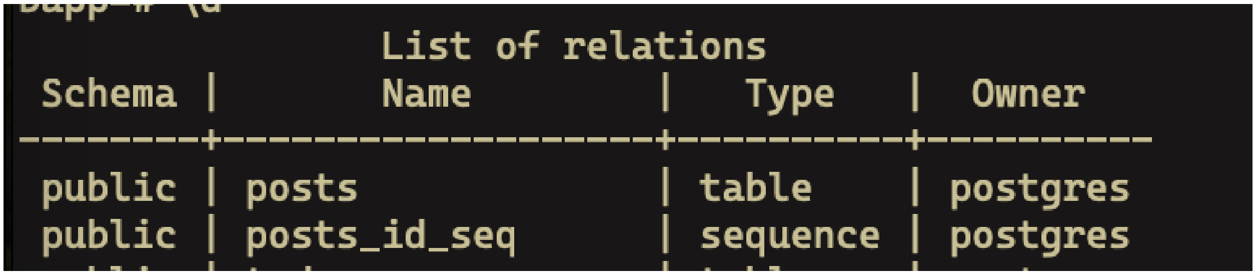
Database:



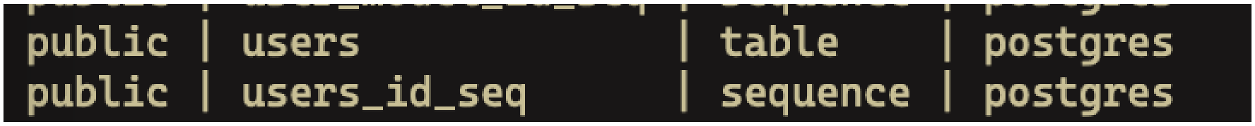
```
bapp=# \d posts
                                           Table "public.posts"
      Column       |          Type          | Collation | Nullable |                Default
-------------------+------------------------+-----------+----------+---------------------------------------
 id                | bigint                 |           | not null | nextval('posts_id_seq'::regclass)
 content_file_name | character varying(255) |           | not null |
 content_type      | character varying(255) |           |          |
 file_name         | character varying(255) |           | not null |
 subtitle          | character varying(255) |           |          |
 title             | character varying(255) |           | not null |
 user_id           | bigint                 |           |          |
Indexes:
    "posts_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "fk5lidm6cqbc7u4xhqpxm898qme" FOREIGN KEY (user_id) REFERENCES users(id)

bapp=# \d users
                                   Table "public.users"
  Column  |          Type          | Collation | Nullable |                Default
----------+------------------------+-----------+----------+---------------------------------------
 id       | bigint                 |           | not null | nextval('users_id_seq'::regclass)
 password | character varying(255) |           | not null |
 role     | character varying(255) |           | not null |
 username | character varying(255) |           | not null |
Indexes:
    "users_pkey" PRIMARY KEY, btree (id)
    "uk_r43af9ap4edm43mmtq01oddj6" UNIQUE CONSTRAINT, btree (username)
Referenced by:
    TABLE "posts" CONSTRAINT "fk5lidm6cqbc7u4xhqpxm898qme" FOREIGN KEY (user_id) REFERENCES users(id)
```



```
bapp=# select * from users;
 id | password |  role  | username
----+----------+--------+----------
  1 | 123      | READER | Achyuta
  2 | 123      | READER | Aditya
(2 rows)

bapp=# select * from posts;
 id | content_file_name |        content_type        |                     file_name                     |  subtitle  |   title    | user_id
----+-------------------+----------------------------+---------------------------------------------------+------------+------------+---------
  3 | admin_123.pem     | application/x-x509-ca-cert | 2d071d59-f17f-438d-87c3-d7820eaaa0ef-admin_123.pem | First Blog | First BLog |       2
(1 row)

bapp=# 
```

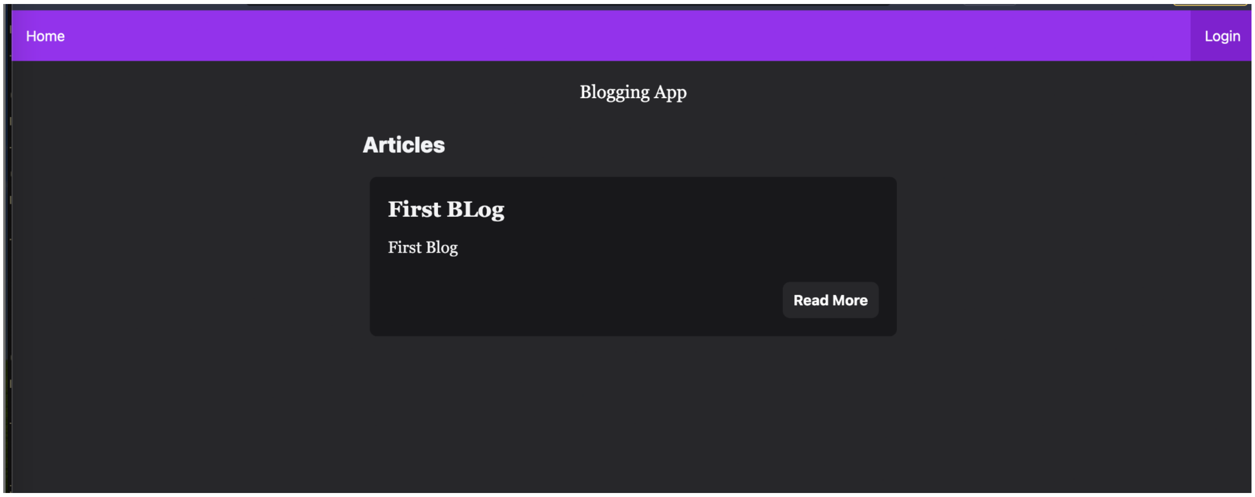


```
bapp=# \d
                 List of relations
 Schema |         Name          |   Type    |  Owner
--------+-----------------------+-----------+----------
 public | posts                 | table     | postgres
 public | posts_id_seq          | sequence  | postgres
```



```
 public | users                 | table     | postgres
 public | users_id_seq          | sequence  | postgres
```

Project Screenshots:

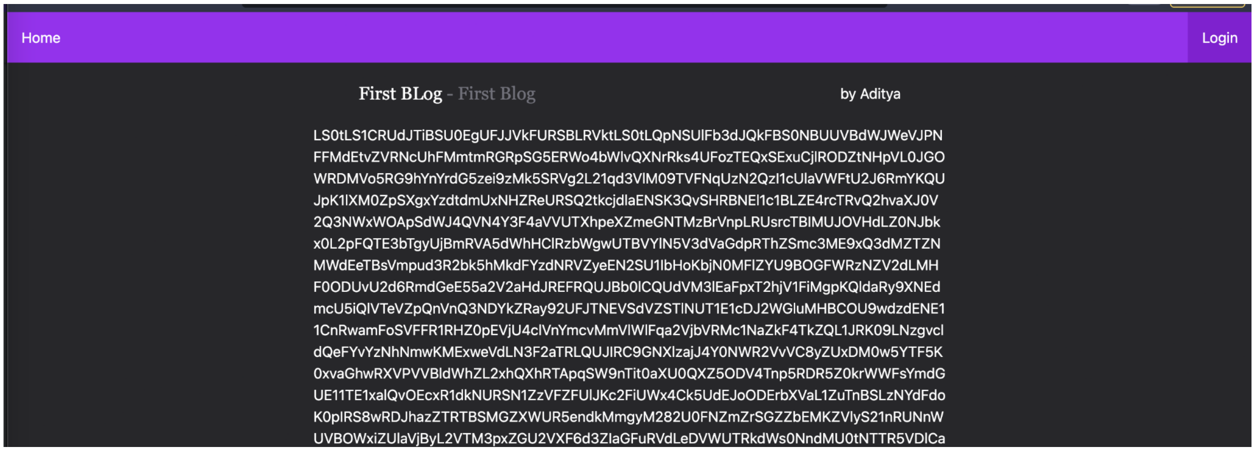


Blogging App

Articles

First BLog

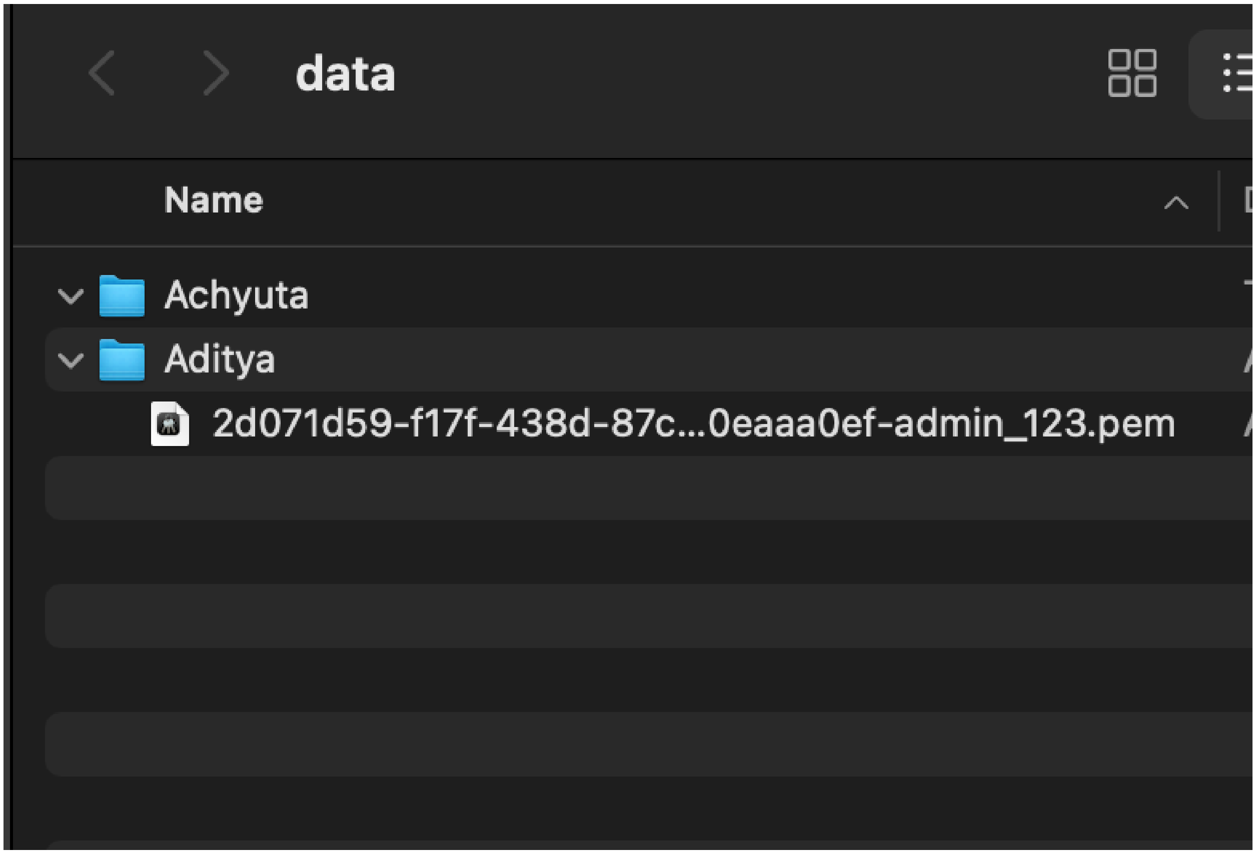
First Blog



First BLog - First Blog

by Aditya

LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFb3dJQkFBS0NBUUVBdWJWeVJPN
FFMdEtvZVRNcUhFMmtmRGRpSG5ERWo4bWlvQXNrRks4UFozTEQxSExuCjlRODZtNHpVL0JGO
WRDMVo5RG9hYnYrdG5zei9zMk5SRVg2L21qd3VlM09TVFNqUzN2QzI1cUlaVWFtU2J6RmYKQU
JpK1lXM0ZpSXgxYzdtdmUxNHZReURSQ2tkcjdlaENSK3QvSHRBNEl1c1BLZE4rcTRvQ2hvaXJ0V
2Q3NWxWOApSdWJ4QVN4Y3F4aVVUXhpeXZmeGNTMzBrVnpLRUsrcTBlMUJOVHdLZ0NJbk
x0L2pFQTE3bTgyUjBmRVA5dWhHClRzbWgwUTBVYlN5V3dVaGdpRThZSmc3ME9xQ3dMZTZN
MWdEeTBsVmpud3R2bk5hMkdFYzdNRVZyeEN2SU1lbHoKbjN0MFlZYU9BOGFWRzNZV2dLMH
F0ODUvU2d6RmdGeE55a2V2aHdJREFRQUJBb0lCQUdVM3lEaFpxT2hjV1FiMgpKQldaRy9XNEd
mcU5iQlVTeVZpQnVnQ3NDYkZRay92UFJTNEVSdVZSTlNUT1E1cDJ2WGluMHBCOU9wdzdENE1
1CnRwamFoSVFFR1RHZ0pEVjU4clVnYmcvMmVlWlFqa2VjbVRMc1NaZkF4TkZQL1JRK09LNzgvcl
dQeFYvYzNhNmwKMExweVdLN3F2aTRLQUJIRC9GNXlzajJ4Y0NWR2VvVC8yZUxDM0w5YTF5K
0xvaGhwRXVPVVBldWhZL2xhQXhRTApqSW9nTit0aXU0QXZ5ODV4Tnp5RDR5Z0krWWFsYmdG
UE11TE1xalQvOEcxR1dkNURSN1ZzVFZFUlJKc2FiUWx4Ck5UdEJoODErbXVaL1ZuTnBSLzNYdFdo
K0plRS8wRDJhazZTRTBSMGZXWUR5endkMmgyM282U0FNZmZrSGZZbEMKZVlyS21nRUNnW
UVBOWxiZUlaVjByL2VTM3pxZGU2VXF6d3ZIaGFuRVdLeDVWUTRkWs0NndMU0tNTTR5VDlCa



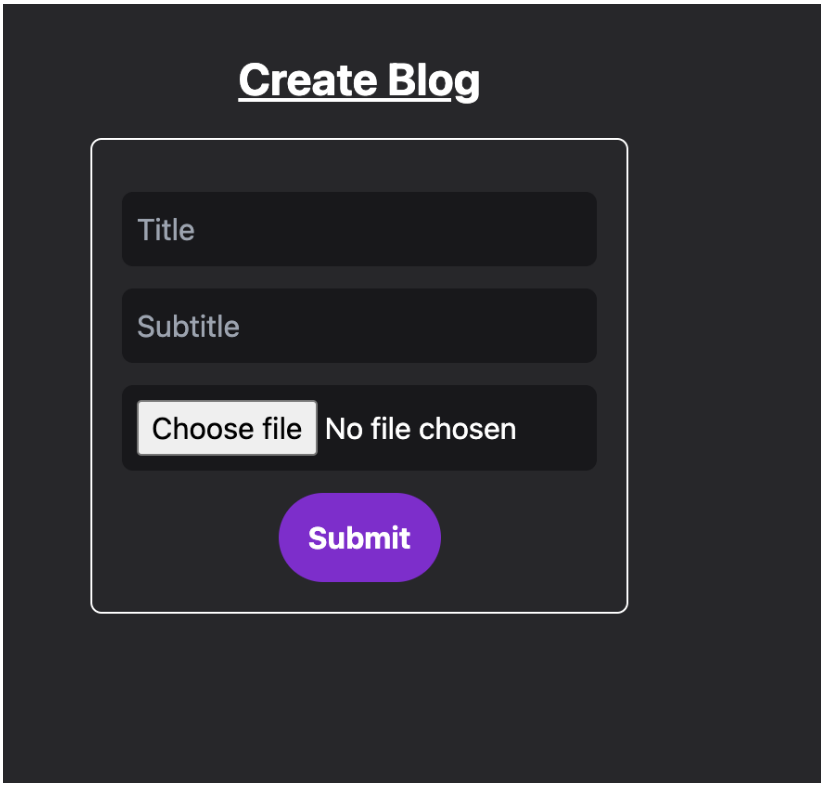
data

Name

Achyuta

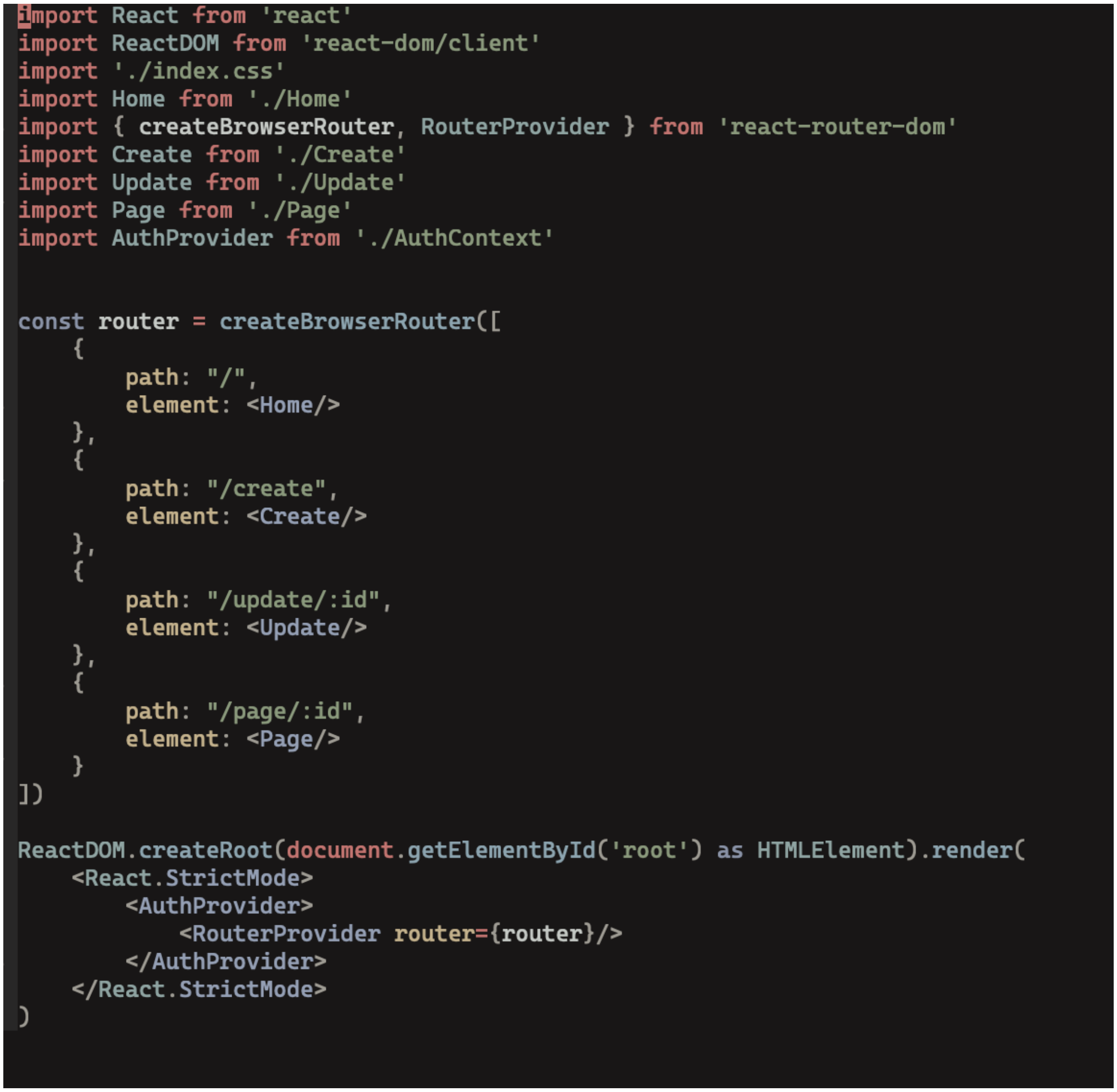
Aditya

2d071d59-f17f-438d-87c...0eaaa0ef-admin_123.pem

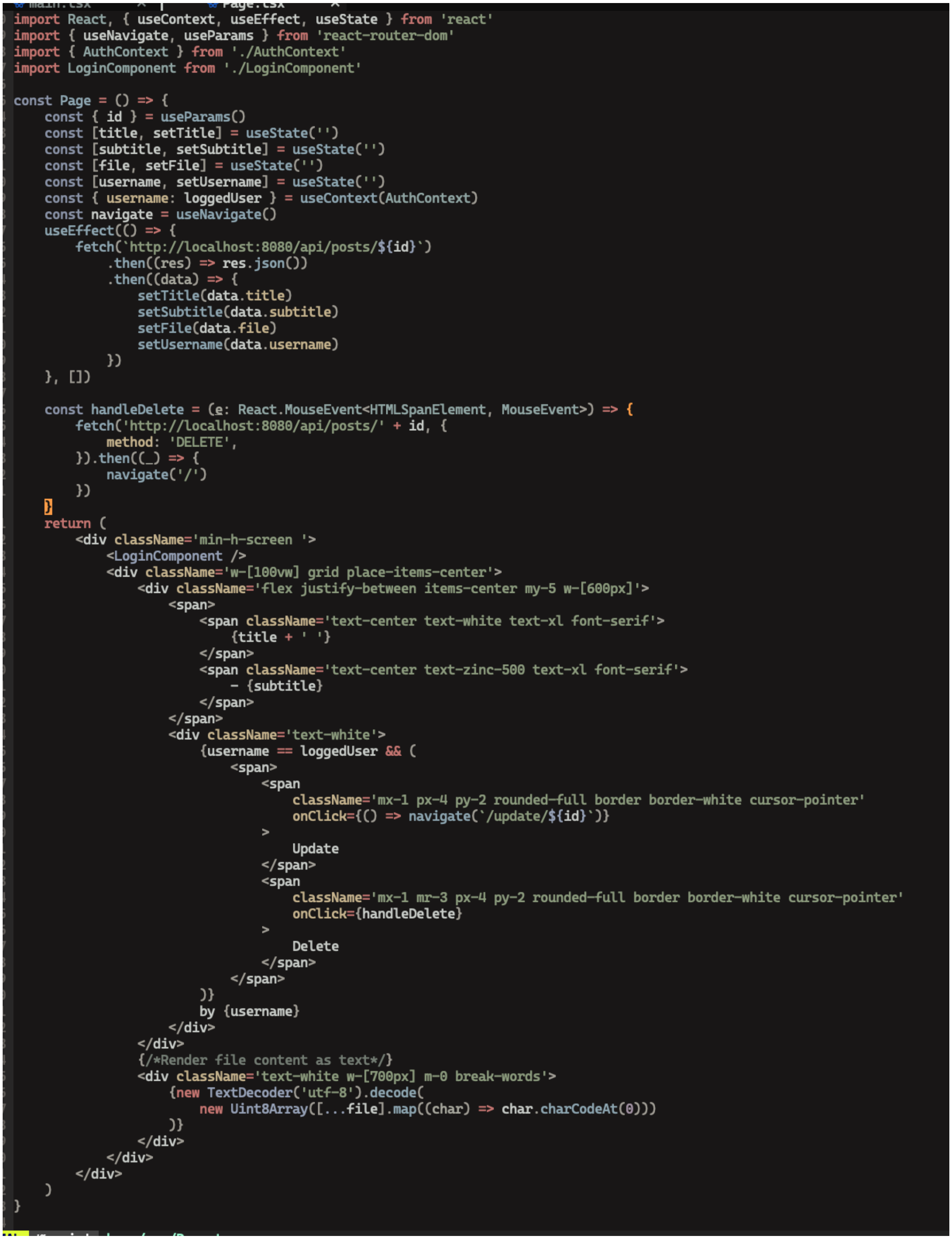


Create Blog

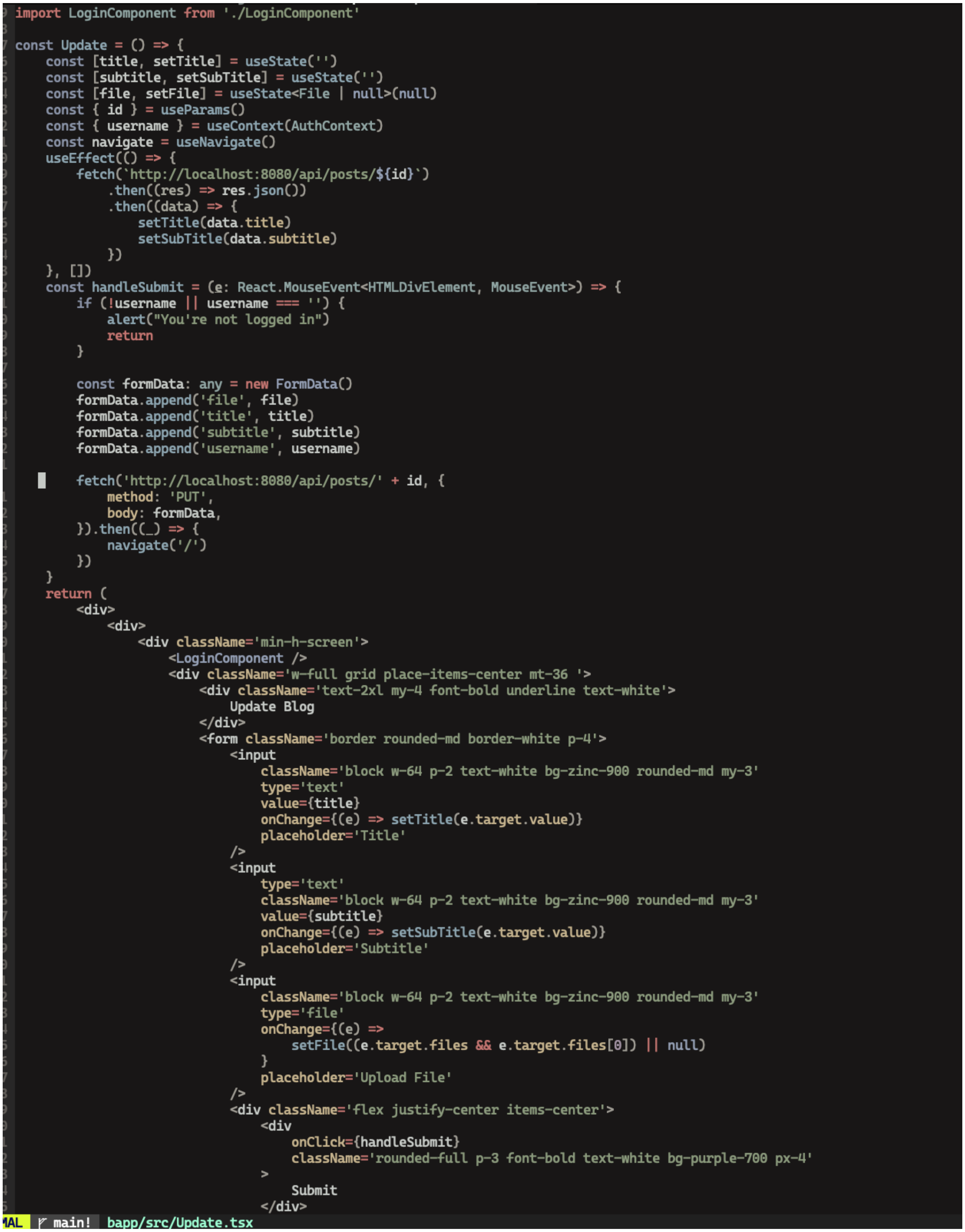
FrontEnd:



```
import React from 'react'
import ReactDOM from 'react-dom/client'
import './index.css'
import Home from './Home'
import { createBrowserRouter, RouterProvider } from 'react-router-dom'
import Create from './Create'
import Update from './Update'
import Page from './Page'
import AuthProvider from './AuthContext'


const router = createBrowserRouter([
    {
        path: "/",
        element: <Home/>
    },
    {
        path: "/create",
        element: <Create/>
    },
    {
        path: "/update/:id",
        element: <Update/>
    },
    {
        path: "/page/:id",
        element: <Page/>
    }
])

ReactDOM.createRoot(document.getElementById('root') as HTMLElement).render(
    <React.StrictMode>
        <AuthProvider>
            <RouterProvider router={router}/>
        </AuthProvider>
    </React.StrictMode>
)
```

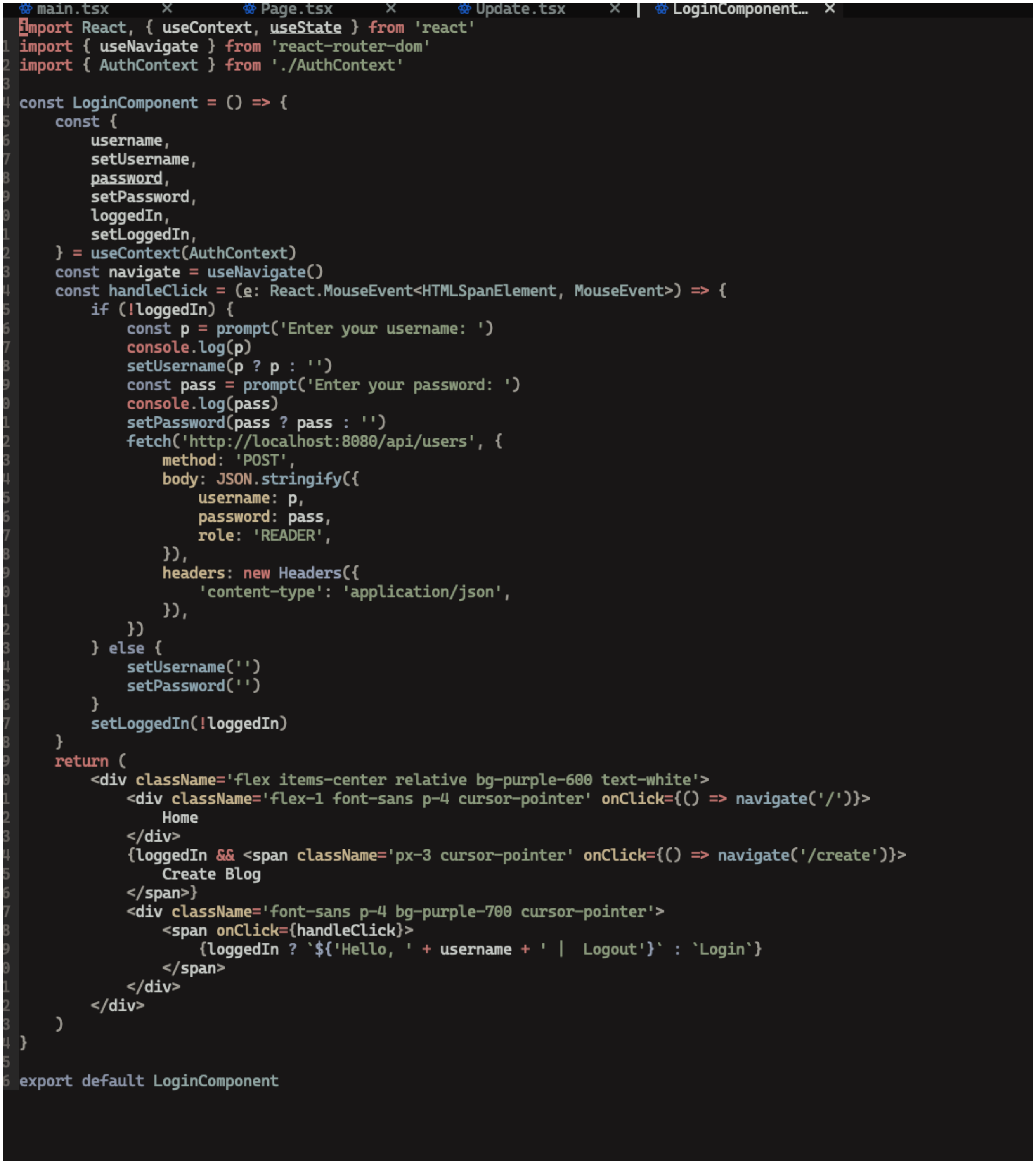


```tsx
import React, { useContext, useEffect, useState } from 'react'
import { useNavigate, useParams } from 'react-router-dom'
import { AuthContext } from './AuthContext'
import LoginComponent from './LoginComponent'

const Page = () => {
    const { id } = useParams()
    const [title, setTitle] = useState('')
    const [subtitle, setSubtitle] = useState('')
    const [file, setFile] = useState('')
    const [username, setUsername] = useState('')
    const { username: loggedUser } = useContext(AuthContext)
    const navigate = useNavigate()
    useEffect(() => {
        fetch(`http://localhost:8080/api/posts/${id}`)
            .then((res) => res.json())
            .then((data) => {
                setTitle(data.title)
                setSubtitle(data.subtitle)
                setFile(data.file)
                setUsername(data.username)
            })
    }, [])

    const handleDelete = (e: React.MouseEvent<HTMLSpanElement, MouseEvent>) => {
        fetch('http://localhost:8080/api/posts/' + id, {
            method: 'DELETE',
        }).then((_) => {
            navigate('/')
        })
    }
    return (
        <div className='min-h-screen '>
            <LoginComponent />
            <div className='w-[100vw] grid place-items-center'>
                <div className='flex justify-between items-center my-5 w-[600px]'>
                    <span>
                        <span className='text-center text-white text-xl font-serif'>
                            {title + ' '}
                        </span>
                        <span className='text-center text-zinc-500 text-xl font-serif'>
                            - {subtitle}
                        </span>
                    </span>
                    <div className='text-white'>
                        {username == loggedUser && (
                            <span>
                                <span
                                    className='mx-1 px-4 py-2 rounded-full border border-white cursor-pointer'
                                    onClick={() => navigate(`/update/${id}`)}
                                >
                                    Update
                                </span>
                                <span
                                    className='mx-1 mr-3 px-4 py-2 rounded-full border border-white cursor-pointer'
                                    onClick={handleDelete}
                                >
                                    Delete
                                </span>
                            </span>
                        )}
                        by {username}
                    </div>
                </div>
                {/*Render file content as text*/}
                <div className='text-white w-[700px] m-0 break-words'>
                    {new TextDecoder('utf-8').decode(
                        new Uint8Array([...file].map((char) => char.charCodeAt(0)))
                    )}
                </div>
            </div>
        </div>
    )
}
```

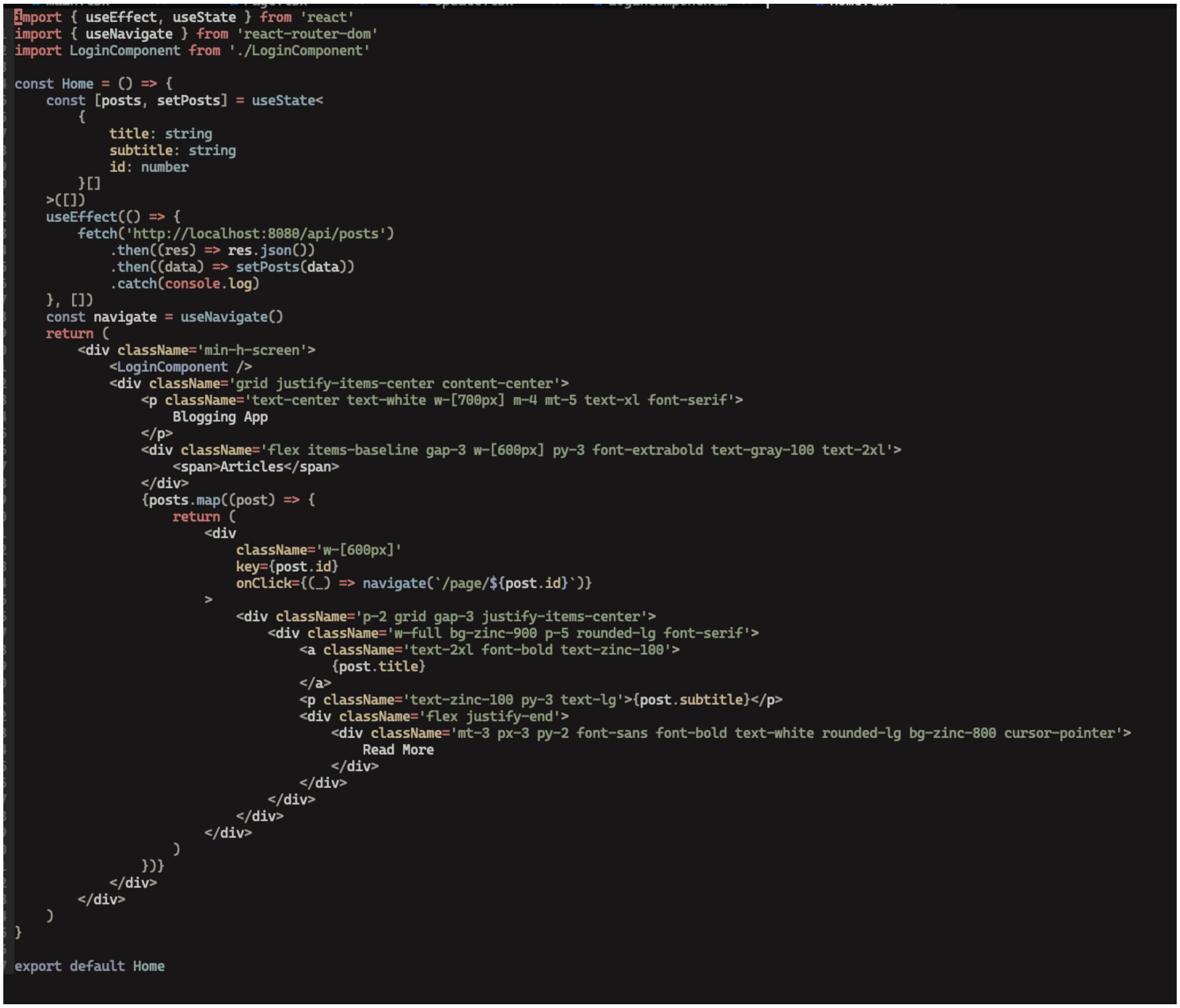


```
import LoginComponent from './LoginComponent'

const Update = () => {
    const [title, setTitle] = useState('')
    const [subtitle, setSubTitle] = useState('')
    const [file, setFile] = useState<File | null>(null)
    const { id } = useParams()
    const { username } = useContext(AuthContext)
    const navigate = useNavigate()
    useEffect(() => {
        fetch(`http://localhost:8080/api/posts/${id}`)
            .then((res) => res.json())
            .then((data) => {
                setTitle(data.title)
                setSubTitle(data.subtitle)
            })
    }, [])
    const handleSubmit = (e: React.MouseEvent<HTMLDivElement, MouseEvent>) => {
        if (!username || username === '') {
            alert("You're not logged in")
            return
        }

        const formData: any = new FormData()
        formData.append('file', file)
        formData.append('title', title)
        formData.append('subtitle', subtitle)
        formData.append('username', username)

        fetch('http://localhost:8080/api/posts/' + id, {
            method: 'PUT',
            body: formData,
        }).then((_) => {
            navigate('/')
        })
    }
    return (
        <div>
            <div>
                <div className='min-h-screen'>
                    <LoginComponent />
                    <div className='w-full grid place-items-center mt-36 '>
                        <div className='text-2xl my-4 font-bold underline text-white'>
                            Update Blog
                        </div>
                        <form className='border rounded-md border-white p-4'>
                            <input
                                className='block w-64 p-2 text-white bg-zinc-900 rounded-md my-3'
                                type='text'
                                value={title}
                                onChange={(e) => setTitle(e.target.value)}
                                placeholder='Title'
                            />
                            <input
                                type='text'
                                className='block w-64 p-2 text-white bg-zinc-900 rounded-md my-3'
                                value={subtitle}
                                onChange={(e) => setSubTitle(e.target.value)}
                                placeholder='Subtitle'
                            />
                            <input
                                className='block w-64 p-2 text-white bg-zinc-900 rounded-md my-3'
                                type='file'
                                onChange={(e) =>
                                    setFile((e.target.files && e.target.files[0]) || null)
                                }
                                placeholder='Upload File'
                            />
                            <div className='flex justify-center items-center'>
                                <div
                                    onClick={handleSubmit}
                                    className='rounded-full p-3 font-bold text-white bg-purple-700 px-4'
                                >
                                    Submit
                                </div>
```

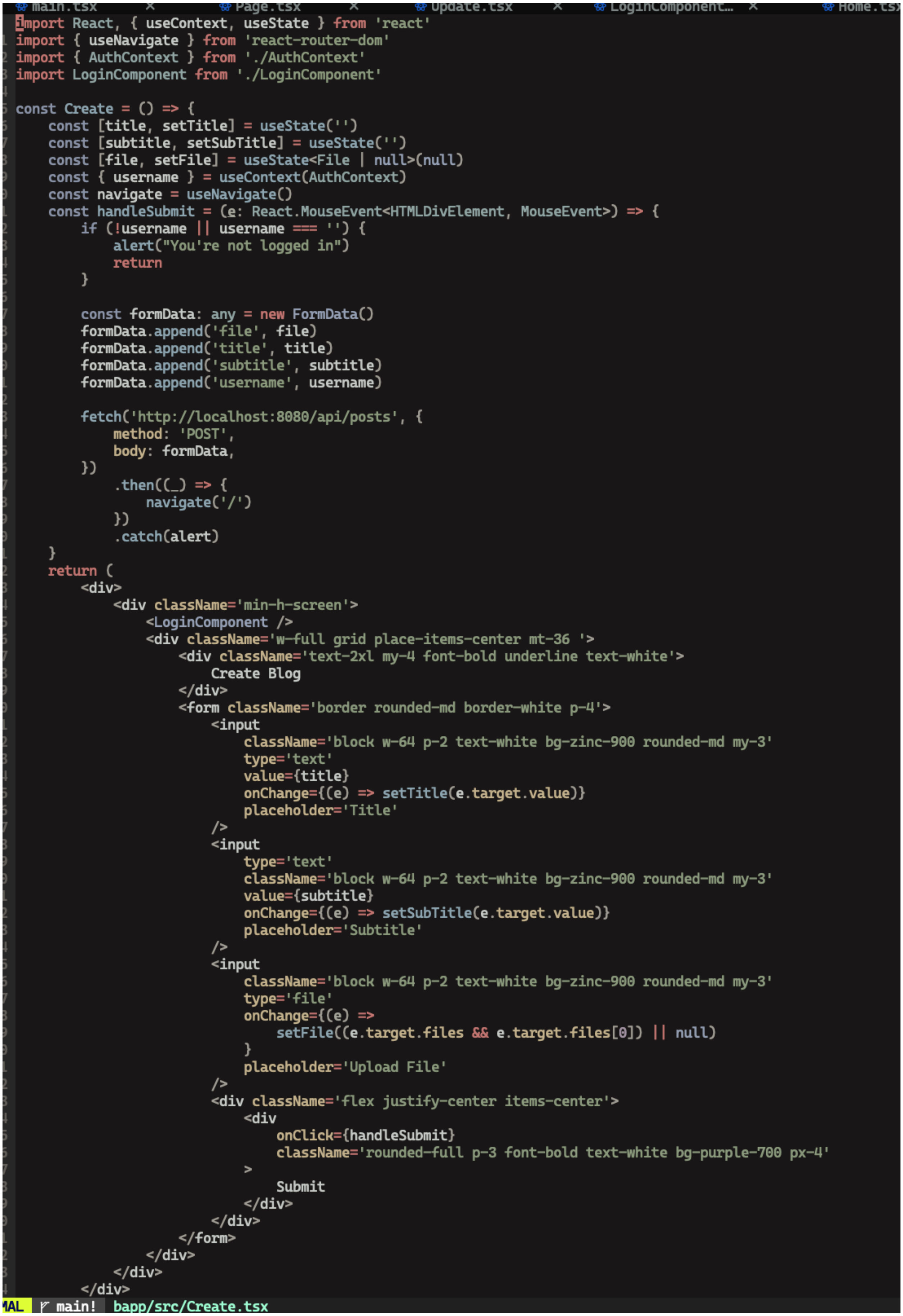
MAL main! bapp/src/Update.tsx



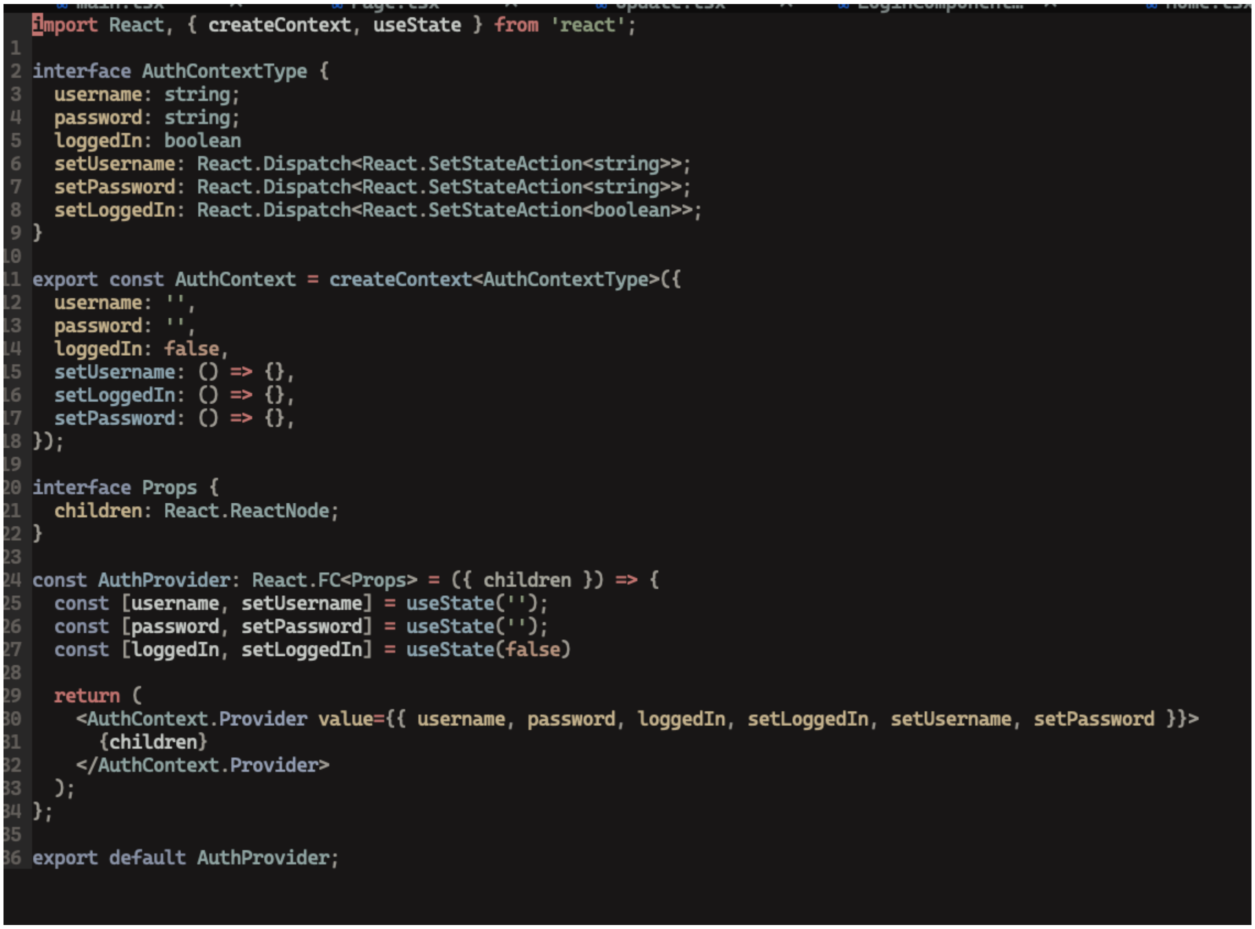
```
import React, { useContext, useState } from 'react'
import { useNavigate } from 'react-router-dom'
import { AuthContext } from './AuthContext'

const LoginComponent = () => {
    const {
        username,
        setUsername,
        password,
        setPassword,
        loggedIn,
        setLoggedIn,
    } = useContext(AuthContext)
    const navigate = useNavigate()
    const handleClick = (e: React.MouseEvent<HTMLSpanElement, MouseEvent>) => {
        if (!loggedIn) {
            const p = prompt('Enter your username: ')
            console.log(p)
            setUsername(p ? p : '')
            const pass = prompt('Enter your password: ')
            console.log(pass)
            setPassword(pass ? pass : '')
            fetch('http://localhost:8080/api/users', {
                method: 'POST',
                body: JSON.stringify({
                    username: p,
                    password: pass,
                    role: 'READER',
                }),
                headers: new Headers({
                    'content-type': 'application/json',
                }),
            })
        } else {
            setUsername('')
            setPassword('')
        }
        setLoggedIn(!loggedIn)
    }
    return (
        <div className='flex items-center relative bg-purple-600 text-white'>
            <div className='flex-1 font-sans p-4 cursor-pointer' onClick={() => navigate('/')}>
                Home
            </div>
            {loggedIn && <span className='px-3 cursor-pointer' onClick={() => navigate('/create')}>
                Create Blog
            </span>}
            <div className='font-sans p-4 bg-purple-700 cursor-pointer'>
                <span onClick={handleClick}>
                    {loggedIn ? `${'Hello, ' + username + ' |  Logout'}` : `Login`}
                </span>
            </div>
        </div>
    )
}

export default LoginComponent
```



```
import { useEffect, useState } from 'react'
import { useNavigate } from 'react-router-dom'
import LoginComponent from './LoginComponent'

const Home = () => {
    const [posts, setPosts] = useState<
        {
            title: string
            subtitle: string
            id: number
        }[]
    >([])
    useEffect(() => {
        fetch('http://localhost:8080/api/posts')
            .then((res) => res.json())
            .then((data) => setPosts(data))
            .catch(console.log)
    }, [])
    const navigate = useNavigate()
    return (
        <div className='min-h-screen'>
            <LoginComponent />
            <div className='grid justify-items-center content-center'>
                <p className='text-center text-white w-[700px] m-4 mt-5 text-xl font-serif'>
                    Blogging App
                </p>
                <div className='flex items-baseline gap-3 w-[600px] py-3 font-extrabold text-gray-100 text-2xl'>
                    <span>Articles</span>
                </div>
                {posts.map((post) => {
                    return (
                        <div
                            className='w-[600px]'
                            key={post.id}
                            onClick={(_) => navigate(`/page/${post.id}`)}
                        >
                            <div className='p-2 grid gap-3 justify-items-center'>
                                <div className='w-full bg-zinc-900 p-5 rounded-lg font-serif'>
                                    <a className='text-2xl font-bold text-zinc-100'>
                                        {post.title}
                                    </a>
                                    <p className='text-zinc-100 py-3 text-lg'>{post.subtitle}</p>
                                    <div className='flex justify-end'>
                                        <div className='mt-3 px-3 py-2 font-sans font-bold text-white rounded-lg bg-zinc-800 cursor-pointer'>
                                            Read More
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    )
                })}
            </div>
        </div>
    )
}

export default Home
```

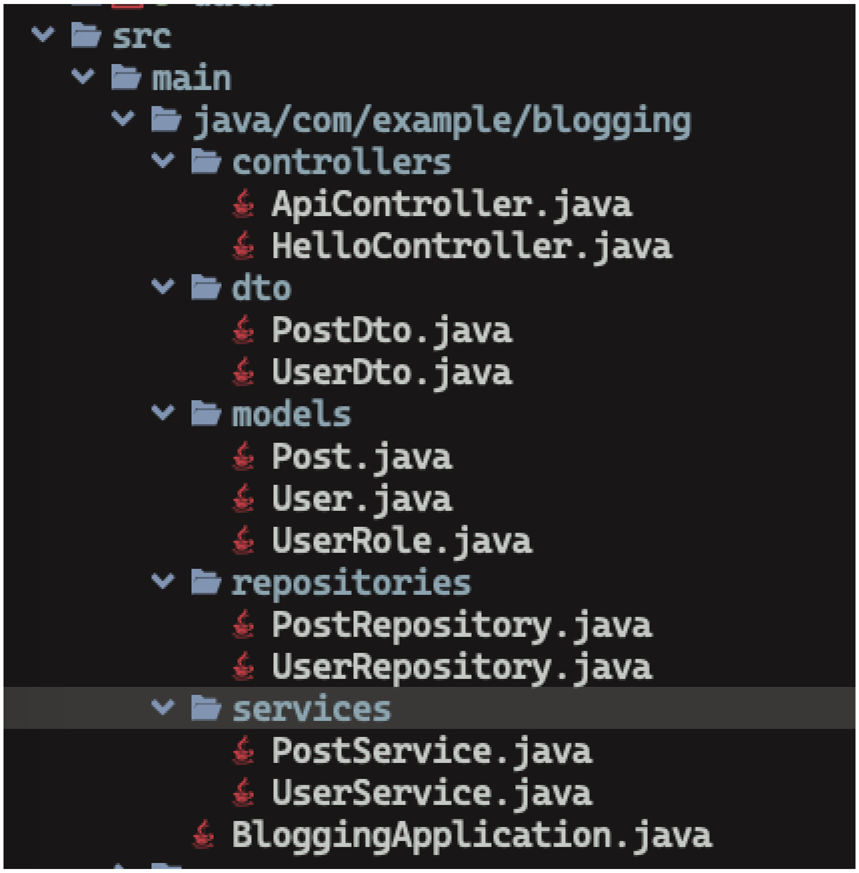


```tsx
import React, { useContext, useState } from 'react'
import { useNavigate } from 'react-router-dom'
import { AuthContext } from './AuthContext'
import LoginComponent from './LoginComponent'

const Create = () => {
    const [title, setTitle] = useState('')
    const [subtitle, setSubTitle] = useState('')
    const [file, setFile] = useState<File | null>(null)
    const { username } = useContext(AuthContext)
    const navigate = useNavigate()
    const handleSubmit = (e: React.MouseEvent<HTMLDivElement, MouseEvent>) => {
        if (!username || username === '') {
            alert("You're not logged in")
            return
        }

        const formData: any = new FormData()
        formData.append('file', file)
        formData.append('title', title)
        formData.append('subtitle', subtitle)
        formData.append('username', username)

        fetch('http://localhost:8080/api/posts', {
            method: 'POST',
            body: formData,
        })
            .then((_) => {
                navigate('/')
            })
            .catch(alert)
    }
    return (
        <div>
            <div className='min-h-screen'>
                <LoginComponent />
                <div className='w-full grid place-items-center mt-36 '>
                    <div className='text-2xl my-4 font-bold underline text-white'>
                        Create Blog
                    </div>
                    <form className='border rounded-md border-white p-4'>
                        <input
                            className='block w-64 p-2 text-white bg-zinc-900 rounded-md my-3'
                            type='text'
                            value={title}
                            onChange={(e) => setTitle(e.target.value)}
                            placeholder='Title'
                        />
                        <input
                            type='text'
                            className='block w-64 p-2 text-white bg-zinc-900 rounded-md my-3'
                            value={subtitle}
                            onChange={(e) => setSubTitle(e.target.value)}
                            placeholder='Subtitle'
                        />
                        <input
                            className='block w-64 p-2 text-white bg-zinc-900 rounded-md my-3'
                            type='file'
                            onChange={(e) =>
                                setFile((e.target.files && e.target.files[0]) || null)
                            }
                            placeholder='Upload File'
                        />
                        <div className='flex justify-center items-center'>
                            <div
                                onClick={handleSubmit}
                                className='rounded-full p-3 font-bold text-white bg-purple-700 px-4'
                            >
                                Submit
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
```

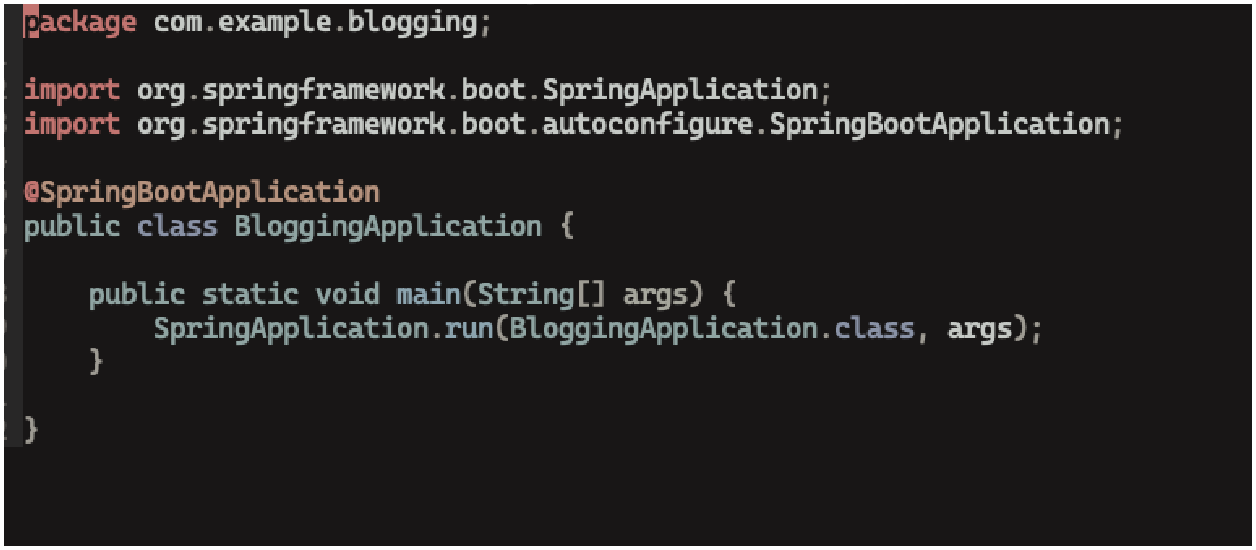


```tsx
import React, { createContext, useState } from 'react';

interface AuthContextType {
  username: string;
  password: string;
  loggedIn: boolean
  setUsername: React.Dispatch<React.SetStateAction<string>>;
  setPassword: React.Dispatch<React.SetStateAction<string>>;
  setLoggedIn: React.Dispatch<React.SetStateAction<boolean>>;
}

export const AuthContext = createContext<AuthContextType>({
  username: '',
  password: '',
  loggedIn: false,
  setUsername: () => {},
  setLoggedIn: () => {},
  setPassword: () => {},
});

interface Props {
  children: React.ReactNode;
}

const AuthProvider: React.FC<Props> = ({ children }) => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [loggedIn, setLoggedIn] = useState(false)

  return (
    <AuthContext.Provider value={{ username, password, loggedIn, setLoggedIn, setUsername, setPassword }}>
      {children}
    </AuthContext.Provider>
  );
};

export default AuthProvider;
```

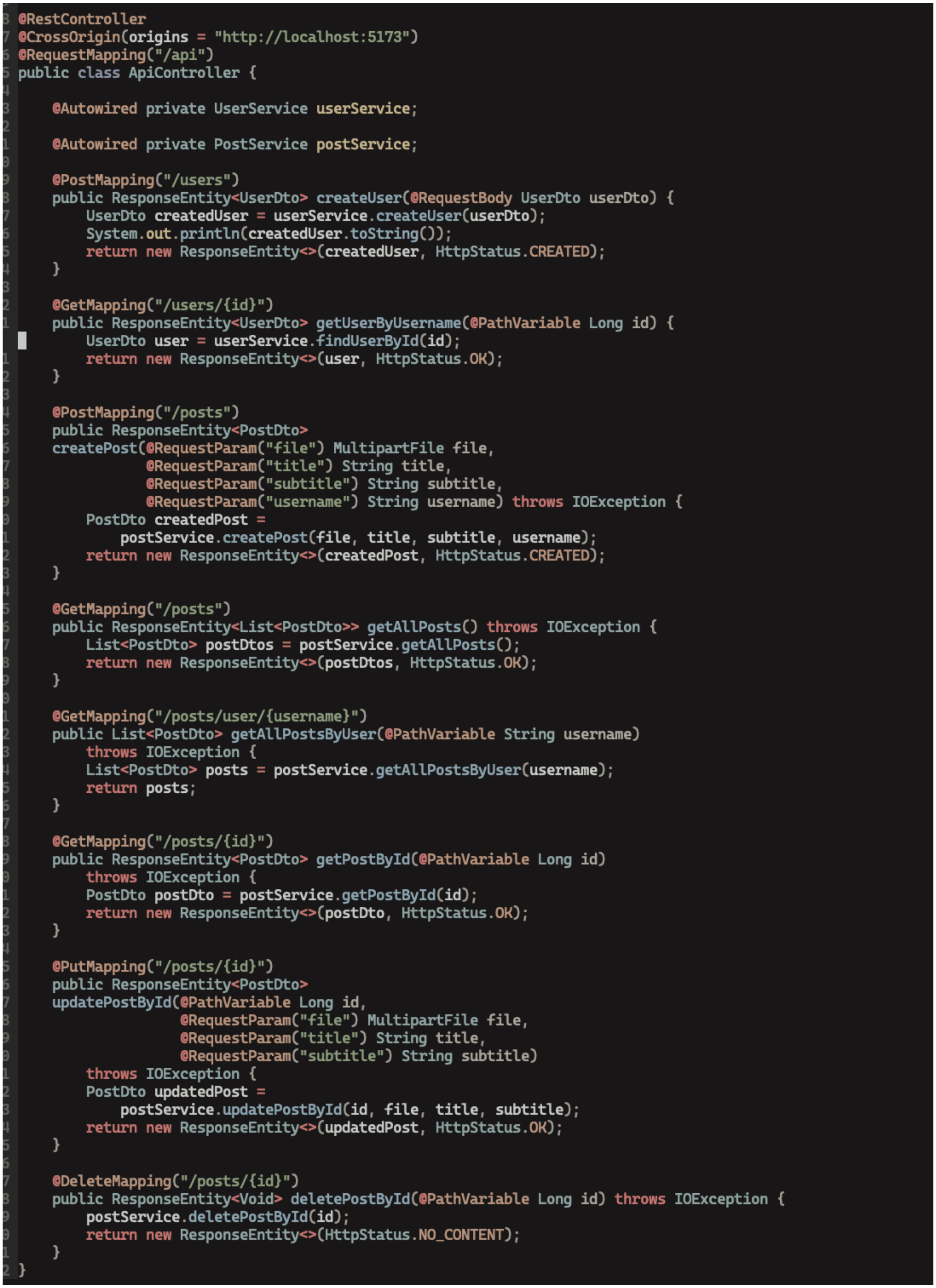
Backend:



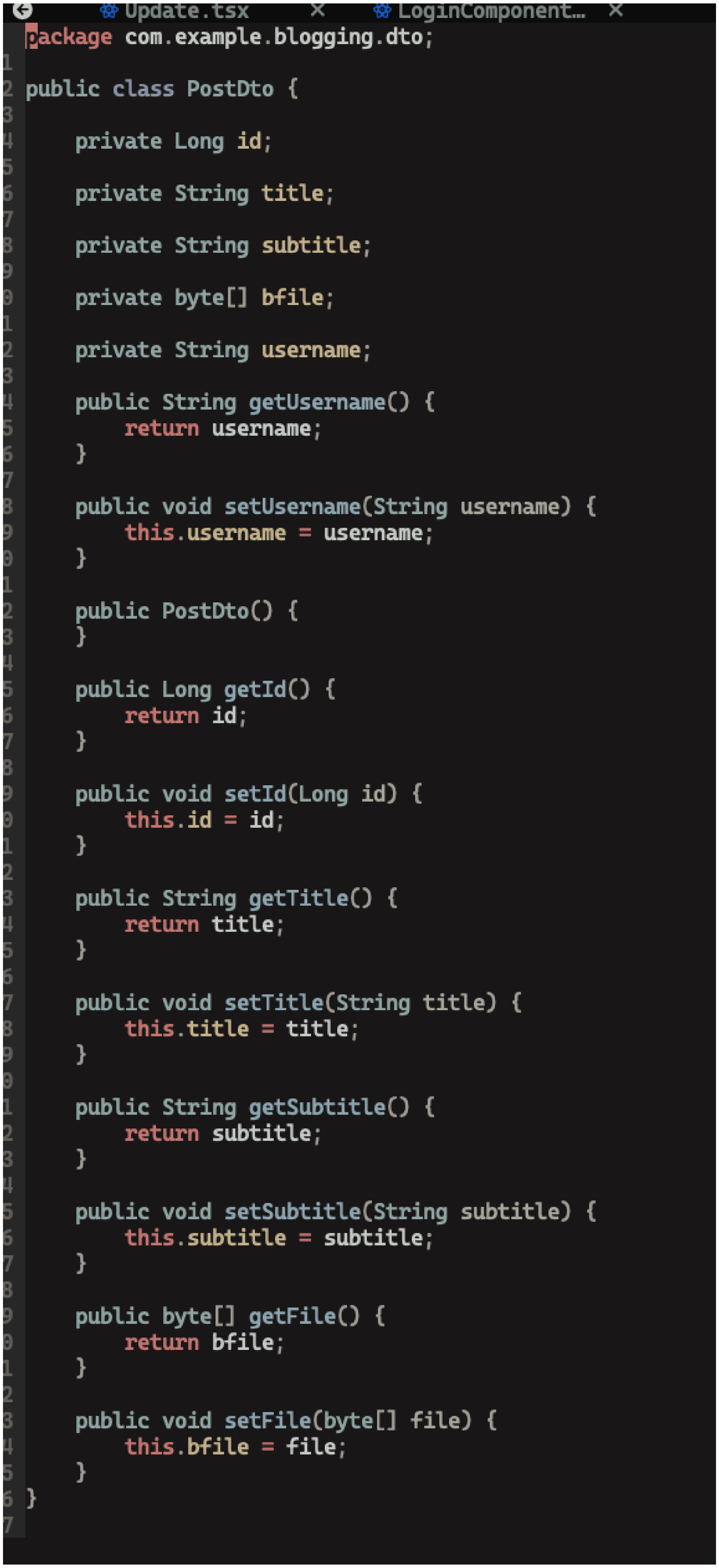
```
src
  main
    java/com/example/blogging
      controllers
        ApiController.java
        HelloController.java
      dto
        PostDto.java
        UserDto.java
      models
        Post.java
        User.java
        UserRole.java
      repositories
        PostRepository.java
        UserRepository.java
      services
        PostService.java
        UserService.java
      BloggingApplication.java
```



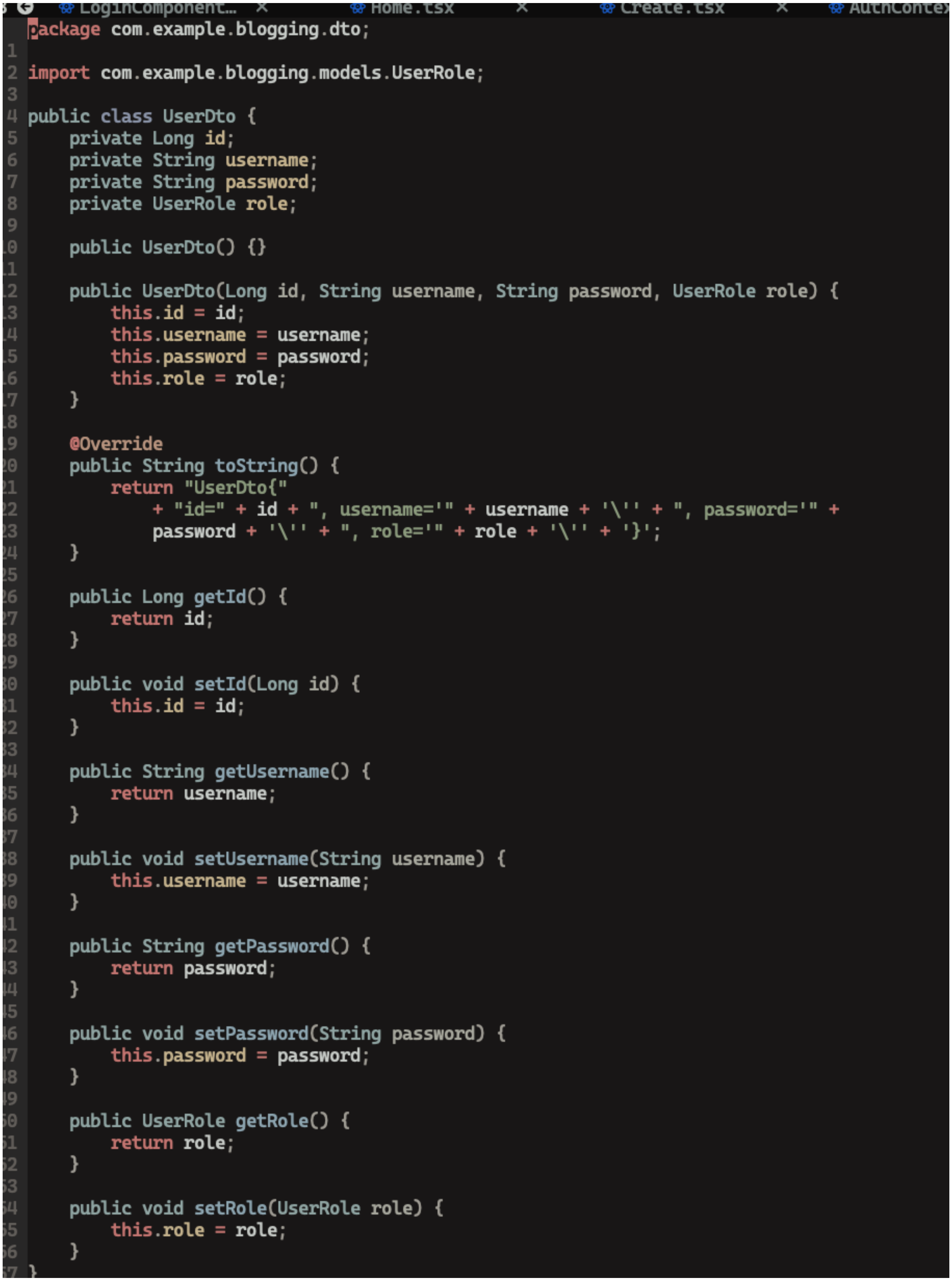
```java
package com.example.blogging;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BloggingApplication {

    public static void main(String[] args) {
        SpringApplication.run(BloggingApplication.class, args);
    }

}
```



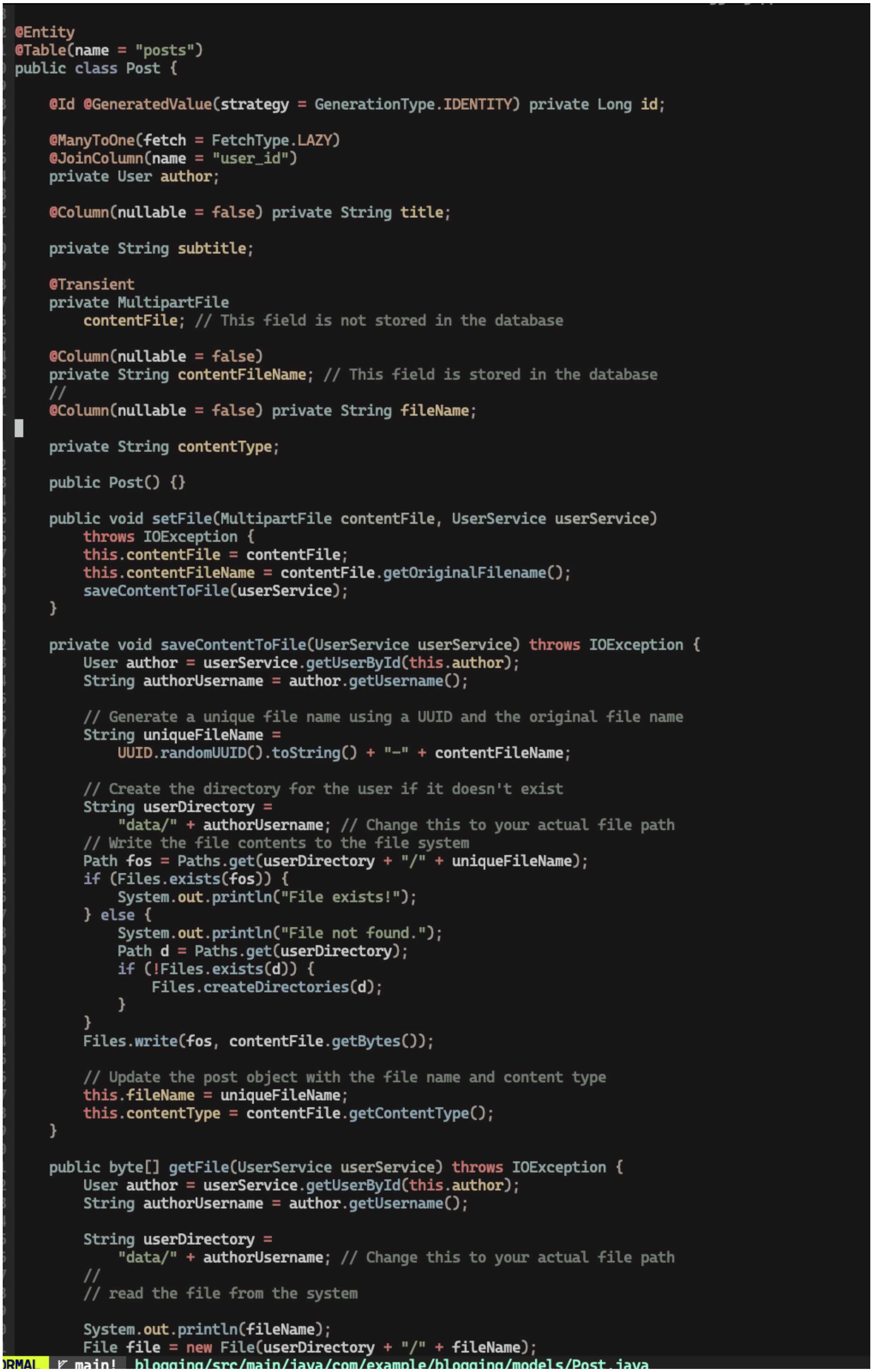
```java
@RestController
@CrossOrigin(origins = "http://localhost:5173")
@RequestMapping("/api")
public class ApiController {

    @Autowired private UserService userService;

    @Autowired private PostService postService;

    @PostMapping("/users")
    public ResponseEntity<UserDto> createUser(@RequestBody UserDto userDto) {
        UserDto createdUser = userService.createUser(userDto);
        System.out.println(createdUser.toString());
        return new ResponseEntity<>(createdUser, HttpStatus.CREATED);
    }

    @GetMapping("/users/{id}")
    public ResponseEntity<UserDto> getUserByUsername(@PathVariable Long id) {
        UserDto user = userService.findUserById(id);
        return new ResponseEntity<>(user, HttpStatus.OK);
    }

    @PostMapping("/posts")
    public ResponseEntity<PostDto>
    createPost(@RequestParam("file") MultipartFile file,
               @RequestParam("title") String title,
               @RequestParam("subtitle") String subtitle,
               @RequestParam("username") String username) throws IOException {
        PostDto createdPost =
            postService.createPost(file, title, subtitle, username);
        return new ResponseEntity<>(createdPost, HttpStatus.CREATED);
    }

    @GetMapping("/posts")
    public ResponseEntity<List<PostDto>> getAllPosts() throws IOException {
        List<PostDto> postDtos = postService.getAllPosts();
        return new ResponseEntity<>(postDtos, HttpStatus.OK);
    }

    @GetMapping("/posts/user/{username}")
    public List<PostDto> getAllPostsByUser(@PathVariable String username)
        throws IOException {
        List<PostDto> posts = postService.getAllPostsByUser(username);
        return posts;
    }

    @GetMapping("/posts/{id}")
    public ResponseEntity<PostDto> getPostById(@PathVariable Long id)
        throws IOException {
        PostDto postDto = postService.getPostById(id);
        return new ResponseEntity<>(postDto, HttpStatus.OK);
    }

    @PutMapping("/posts/{id}")
    public ResponseEntity<PostDto>
    updatePostById(@PathVariable Long id,
                   @RequestParam("file") MultipartFile file,
                   @RequestParam("title") String title,
                   @RequestParam("subtitle") String subtitle)
        throws IOException {
        PostDto updatedPost =
            postService.updatePostById(id, file, title, subtitle);
        return new ResponseEntity<>(updatedPost, HttpStatus.OK);
    }

    @DeleteMapping("/posts/{id}")
    public ResponseEntity<Void> deletePostById(@PathVariable Long id) throws IOException {
        postService.deletePostById(id);
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    }
}
```



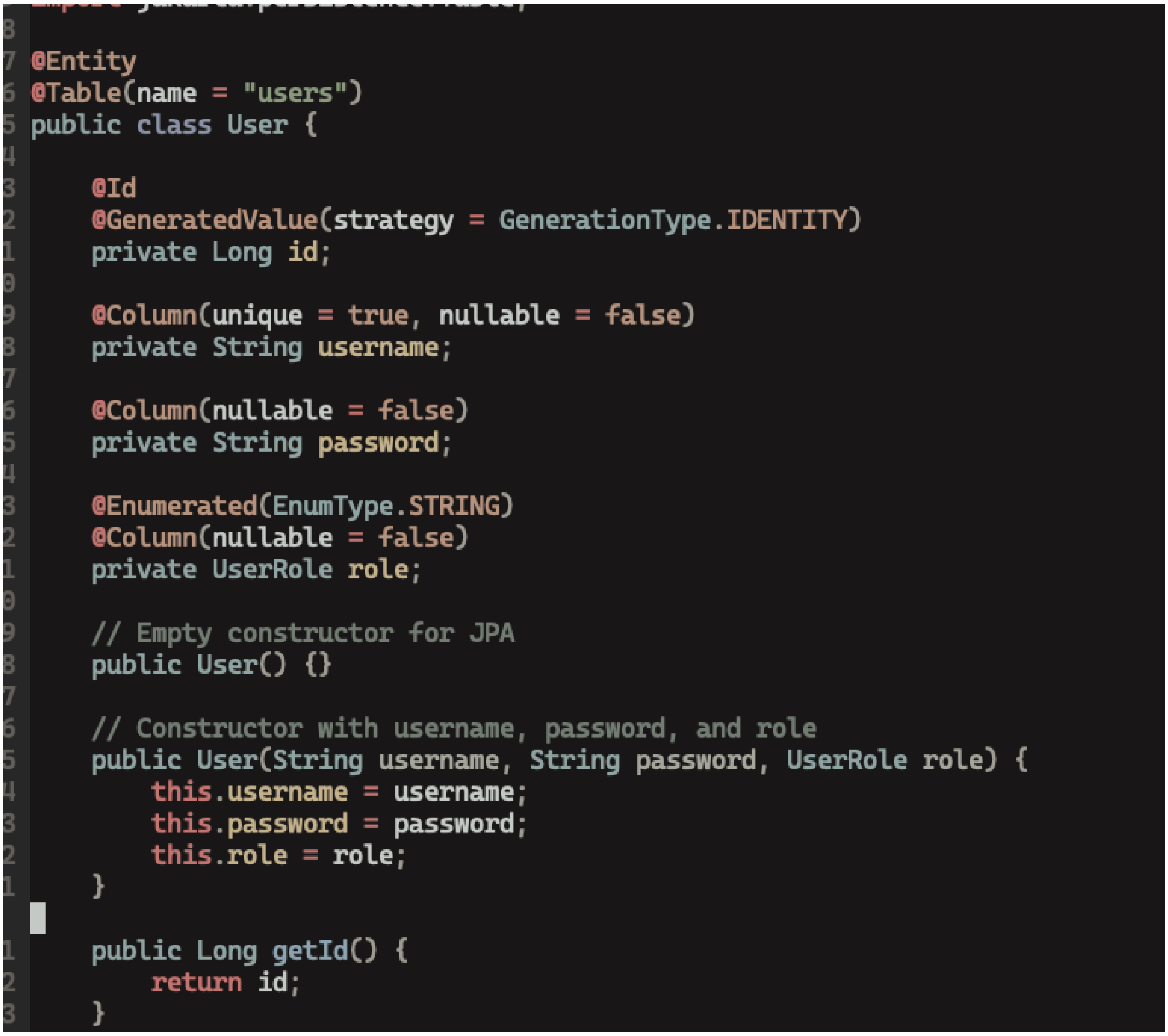
```java
package com.example.blogging.dto;

public class PostDto {

    private Long id;

    private String title;

    private String subtitle;

    private byte[] bfile;

    private String username;

    public String getUsername() {
        return username;
    }

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

    public PostDto() {
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getSubtitle() {
        return subtitle;
    }

    public void setSubtitle(String subtitle) {
        this.subtitle = subtitle;
    }

    public byte[] getFile() {
        return bfile;
    }

    public void setFile(byte[] file) {
        this.bfile = file;
    }
}
```



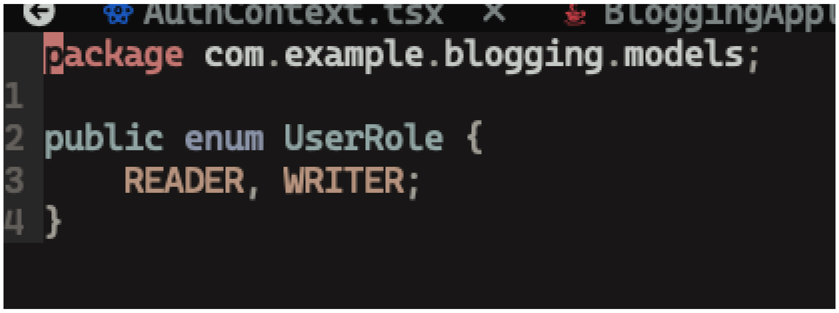
```java
package com.example.blogging.dto;

import com.example.blogging.models.UserRole;

public class UserDto {
    private Long id;
    private String username;
    private String password;
    private UserRole role;

    public UserDto() {}

    public UserDto(Long id, String username, String password, UserRole role) {
        this.id = id;
        this.username = username;
        this.password = password;
        this.role = role;
    }

    @Override
    public String toString() {
        return "UserDto{"
            + "id=" + id + ", username='" + username + '\'' + ", password='" +
            password + '\'' + ", role='" + role + '\'' + '}';
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

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

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public UserRole getRole() {
        return role;
    }

    public void setRole(UserRole role) {
        this.role = role;
    }
}
```



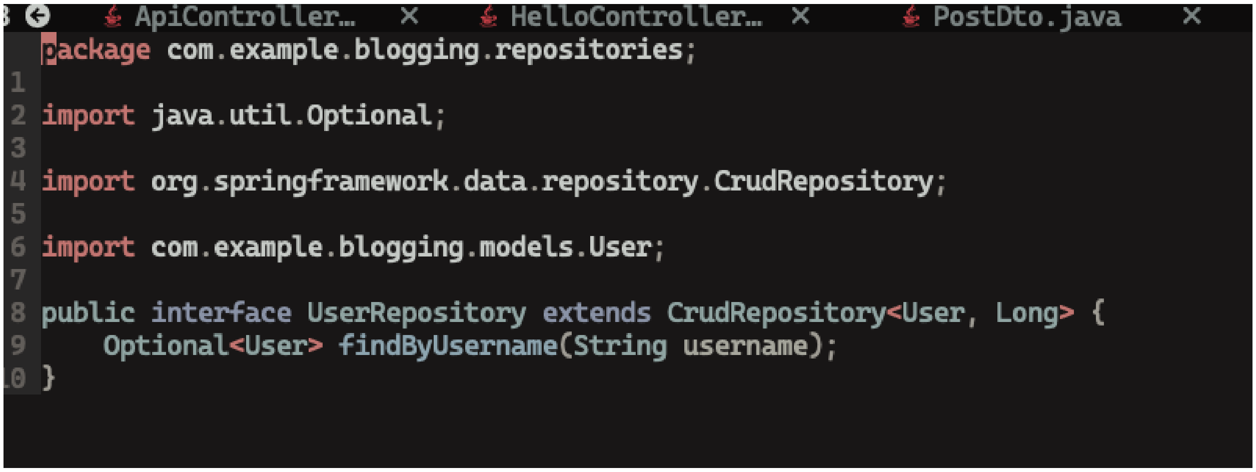
```java
@Entity
@Table(name = "posts")
public class Post {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY) private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id")
    private User author;

    @Column(nullable = false) private String title;

    private String subtitle;

    @Transient
    private MultipartFile
        contentFile; // This field is not stored in the database

    @Column(nullable = false)
    private String contentFileName; // This field is stored in the database
    //
    @Column(nullable = false) private String fileName;

    private String contentType;

    public Post() {}

    public void setFile(MultipartFile contentFile, UserService userService)
        throws IOException {
        this.contentFile = contentFile;
        this.contentFileName = contentFile.getOriginalFilename();
        saveContentToFile(userService);
    }

    private void saveContentToFile(UserService userService) throws IOException {
        User author = userService.getUserById(this.author);
        String authorUsername = author.getUsername();

        // Generate a unique file name using a UUID and the original file name
        String uniqueFileName =
            UUID.randomUUID().toString() + "-" + contentFileName;

        // Create the directory for the user if it doesn't exist
        String userDirectory =
            "data/" + authorUsername; // Change this to your actual file path
        // Write the file contents to the file system
        Path fos = Paths.get(userDirectory + "/" + uniqueFileName);
        if (Files.exists(fos)) {
            System.out.println("File exists!");
        } else {
            System.out.println("File not found.");
            Path d = Paths.get(userDirectory);
            if (!Files.exists(d)) {
                Files.createDirectories(d);
            }
        }
        Files.write(fos, contentFile.getBytes());

        // Update the post object with the file name and content type
        this.fileName = uniqueFileName;
        this.contentType = contentFile.getContentType();
    }

    public byte[] getFile(UserService userService) throws IOException {
        User author = userService.getUserById(this.author);
        String authorUsername = author.getUsername();

        String userDirectory =
            "data/" + authorUsername; // Change this to your actual file path
        //
        // read the file from the system

        System.out.println(fileName);
        File file = new File(userDirectory + "/" + fileName);
ORMAL  main!  blogging/src/main/java/com/example/blogging/models/Post.java
```



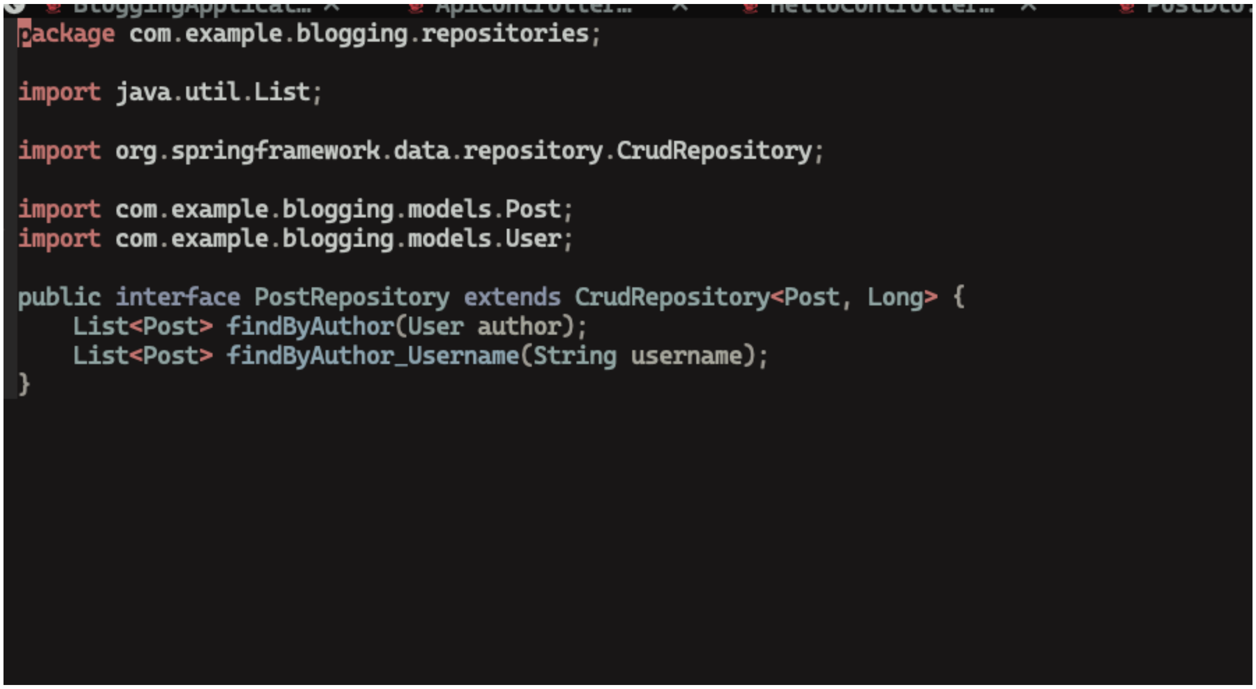
```java
import jakarta.persistence.Table;

@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true, nullable = false)
    private String username;

    @Column(nullable = false)
    private String password;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private UserRole role;

    // Empty constructor for JPA
    public User() {}

    // Constructor with username, password, and role
    public User(String username, String password, UserRole role) {
        this.username = username;
        this.password = password;
        this.role = role;
    }

    public Long getId() {
        return id;
    }
```



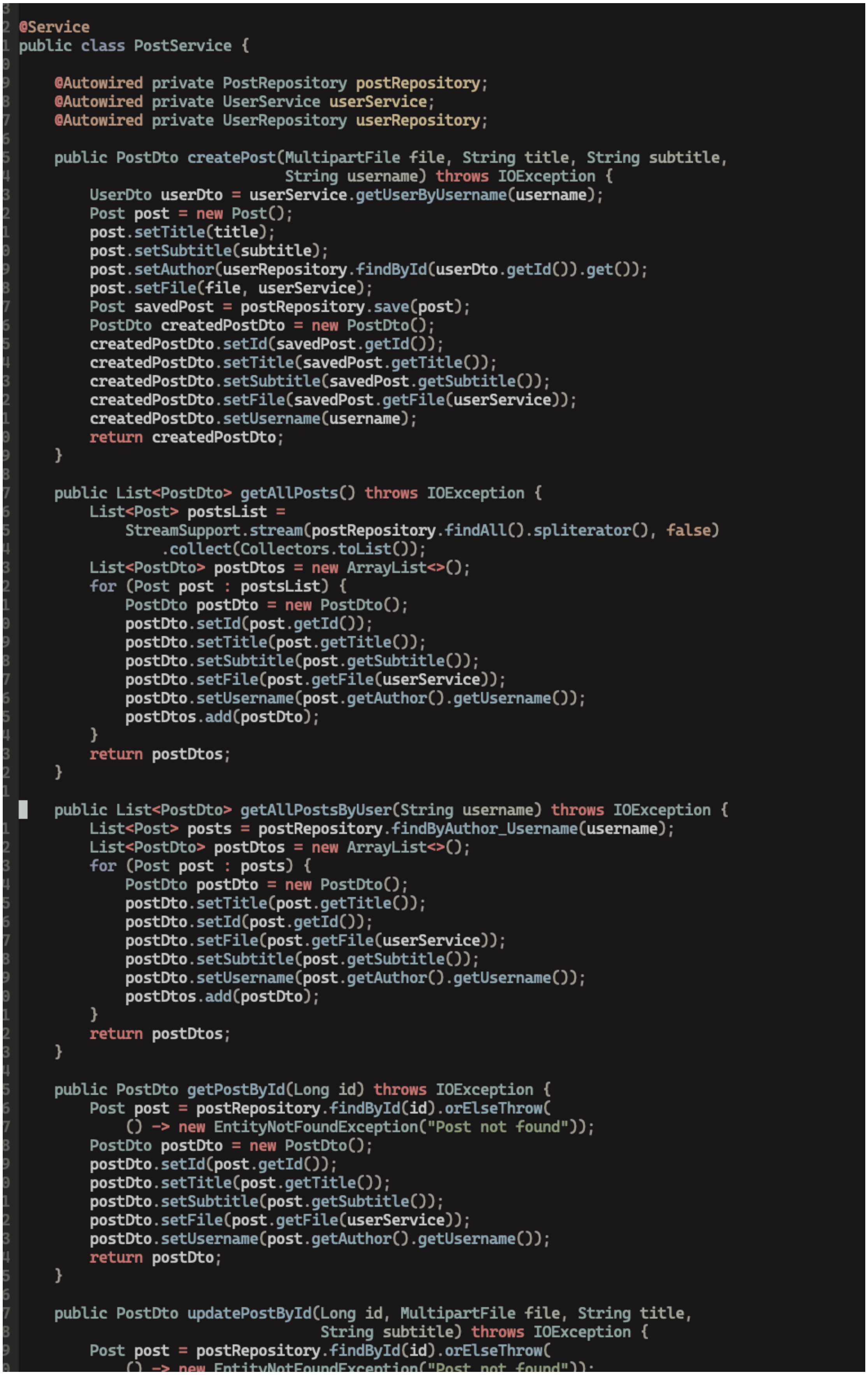
```java
package com.example.blogging.models;

public enum UserRole {
    READER, WRITER;
}
```



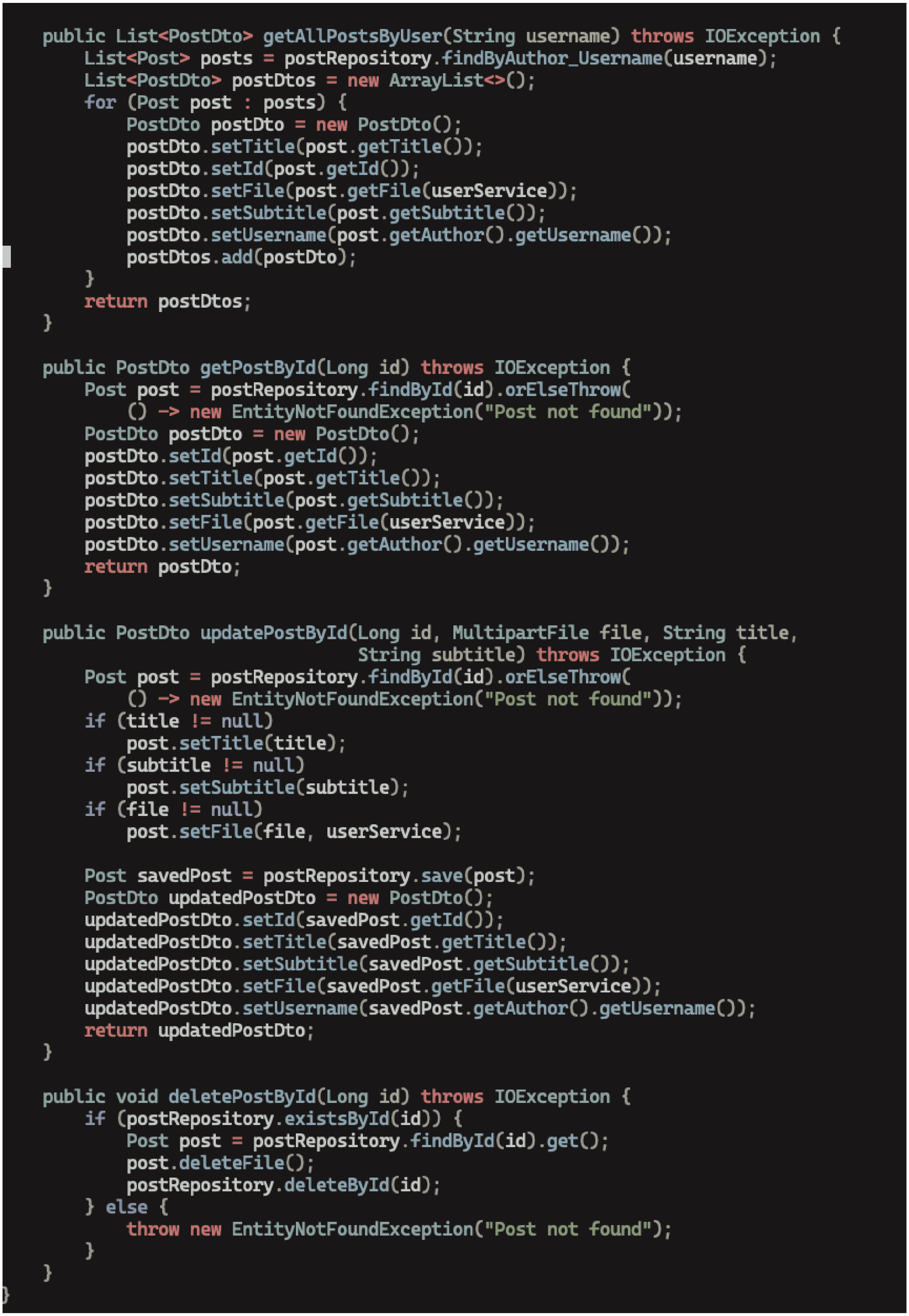
```java
package com.example.blogging.repositories;

import java.util.Optional;

import org.springframework.data.repository.CrudRepository;

import com.example.blogging.models.User;

public interface UserRepository extends CrudRepository<User, Long> {
    Optional<User> findByUsername(String username);
}
```



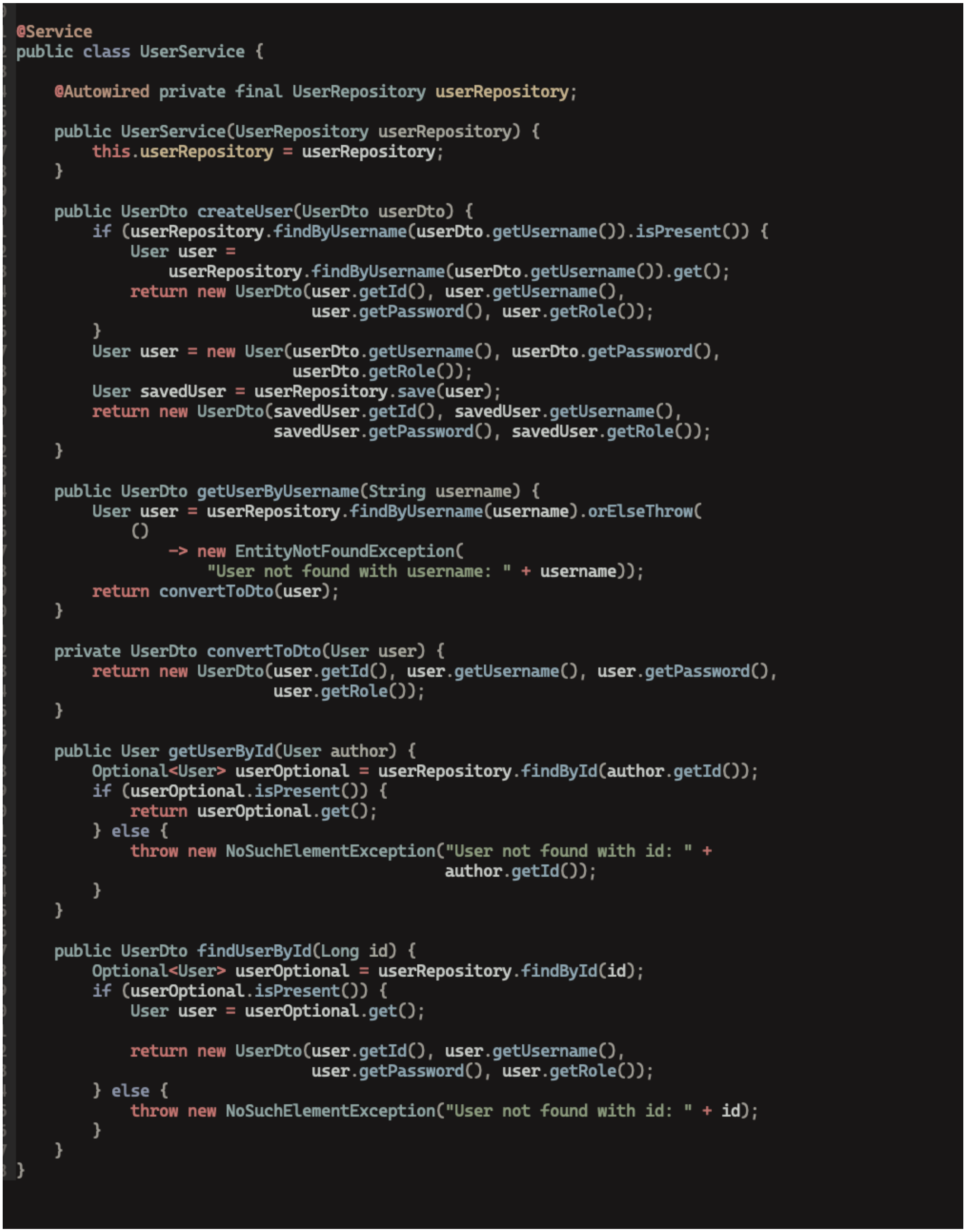
```java
package com.example.blogging.repositories;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import com.example.blogging.models.Post;
import com.example.blogging.models.User;

public interface PostRepository extends CrudRepository<Post, Long> {
    List<Post> findByAuthor(User author);
    List<Post> findByAuthor_Username(String username);
}
```



```
@Service
public class PostService {

    @Autowired private PostRepository postRepository;
    @Autowired private UserService userService;
    @Autowired private UserRepository userRepository;

    public PostDto createPost(MultipartFile file, String title, String subtitle,
                              String username) throws IOException {
        UserDto userDto = userService.getUserByUsername(username);
        Post post = new Post();
        post.setTitle(title);
        post.setSubtitle(subtitle);
        post.setAuthor(userRepository.findById(userDto.getId()).get());
        post.setFile(file, userService);
        Post savedPost = postRepository.save(post);
        PostDto createdPostDto = new PostDto();
        createdPostDto.setId(savedPost.getId());
        createdPostDto.setTitle(savedPost.getTitle());
        createdPostDto.setSubtitle(savedPost.getSubtitle());
        createdPostDto.setFile(savedPost.getFile(userService));
        createdPostDto.setUsername(username);
        return createdPostDto;
    }

    public List<PostDto> getAllPosts() throws IOException {
        List<Post> postsList =
            StreamSupport.stream(postRepository.findAll().spliterator(), false)
                .collect(Collectors.toList());
        List<PostDto> postDtos = new ArrayList<>();
        for (Post post : postsList) {
            PostDto postDto = new PostDto();
            postDto.setId(post.getId());
            postDto.setTitle(post.getTitle());
            postDto.setSubtitle(post.getSubtitle());
            postDto.setFile(post.getFile(userService));
            postDto.setUsername(post.getAuthor().getUsername());
            postDtos.add(postDto);
        }
        return postDtos;
    }

    public List<PostDto> getAllPostsByUser(String username) throws IOException {
        List<Post> posts = postRepository.findByAuthor_Username(username);
        List<PostDto> postDtos = new ArrayList<>();
        for (Post post : posts) {
            PostDto postDto = new PostDto();
            postDto.setTitle(post.getTitle());
            postDto.setId(post.getId());
            postDto.setFile(post.getFile(userService));
            postDto.setSubtitle(post.getSubtitle());
            postDto.setUsername(post.getAuthor().getUsername());
            postDtos.add(postDto);
        }
        return postDtos;
    }

    public PostDto getPostById(Long id) throws IOException {
        Post post = postRepository.findById(id).orElseThrow(
            () -> new EntityNotFoundException("Post not found"));
        PostDto postDto = new PostDto();
        postDto.setId(post.getId());
        postDto.setTitle(post.getTitle());
        postDto.setSubtitle(post.getSubtitle());
        postDto.setFile(post.getFile(userService));
        postDto.setUsername(post.getAuthor().getUsername());
        return postDto;
    }

    public PostDto updatePostById(Long id, MultipartFile file, String title,
                                  String subtitle) throws IOException {
        Post post = postRepository.findById(id).orElseThrow(
            () -> new EntityNotFoundException("Post not found"));
```



```java
    public List<PostDto> getAllPostsByUser(String username) throws IOException {
        List<Post> posts = postRepository.findByAuthor_Username(username);
        List<PostDto> postDtos = new ArrayList<>();
        for (Post post : posts) {
            PostDto postDto = new PostDto();
            postDto.setTitle(post.getTitle());
            postDto.setId(post.getId());
            postDto.setFile(post.getFile(userService));
            postDto.setSubtitle(post.getSubtitle());
            postDto.setUsername(post.getAuthor().getUsername());
            postDtos.add(postDto);
        }
        return postDtos;
    }

    public PostDto getPostById(Long id) throws IOException {
        Post post = postRepository.findById(id).orElseThrow(
            () -> new EntityNotFoundException("Post not found"));
        PostDto postDto = new PostDto();
        postDto.setId(post.getId());
        postDto.setTitle(post.getTitle());
        postDto.setSubtitle(post.getSubtitle());
        postDto.setFile(post.getFile(userService));
        postDto.setUsername(post.getAuthor().getUsername());
        return postDto;
    }

    public PostDto updatePostById(Long id, MultipartFile file, String title,
                                  String subtitle) throws IOException {
        Post post = postRepository.findById(id).orElseThrow(
            () -> new EntityNotFoundException("Post not found"));
        if (title != null)
            post.setTitle(title);
        if (subtitle != null)
            post.setSubtitle(subtitle);
        if (file != null)
            post.setFile(file, userService);

        Post savedPost = postRepository.save(post);
        PostDto updatedPostDto = new PostDto();
        updatedPostDto.setId(savedPost.getId());
        updatedPostDto.setTitle(savedPost.getTitle());
        updatedPostDto.setSubtitle(savedPost.getSubtitle());
        updatedPostDto.setFile(savedPost.getFile(userService));
        updatedPostDto.setUsername(savedPost.getAuthor().getUsername());
        return updatedPostDto;
    }

    public void deletePostById(Long id) throws IOException {
        if (postRepository.existsById(id)) {
            Post post = postRepository.findById(id).get();
            post.deleteFile();
            postRepository.deleteById(id);
        } else {
            throw new EntityNotFoundException("Post not found");
        }
    }
}
```



```java
@Service
public class UserService {

    @Autowired private final UserRepository userRepository;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public UserDto createUser(UserDto userDto) {
        if (userRepository.findByUsername(userDto.getUsername()).isPresent()) {
            User user =
                userRepository.findByUsername(userDto.getUsername()).get();
            return new UserDto(user.getId(), user.getUsername(),
                               user.getPassword(), user.getRole());
        }
        User user = new User(userDto.getUsername(), userDto.getPassword(),
                             userDto.getRole());
        User savedUser = userRepository.save(user);
        return new UserDto(savedUser.getId(), savedUser.getUsername(),
                           savedUser.getPassword(), savedUser.getRole());
    }

    public UserDto getUserByUsername(String username) {
        User user = userRepository.findByUsername(username).orElseThrow(
            ()
                -> new EntityNotFoundException(
                    "User not found with username: " + username));
        return convertToDto(user);
    }

    private UserDto convertToDto(User user) {
        return new UserDto(user.getId(), user.getUsername(), user.getPassword(),
                           user.getRole());
    }

    public User getUserById(User author) {
        Optional<User> userOptional = userRepository.findById(author.getId());
        if (userOptional.isPresent()) {
            return userOptional.get();
        } else {
            throw new NoSuchElementException("User not found with id: " +
                                             author.getId());
        }
    }

    public UserDto findUserById(Long id) {
        Optional<User> userOptional = userRepository.findById(id);
        if (userOptional.isPresent()) {
            User user = userOptional.get();

            return new UserDto(user.getId(), user.getUsername(),
                               user.getPassword(), user.getRole());
        } else {
            throw new NoSuchElementException("User not found with id: " + id);
        }
    }
}
```