

Kerekegyensúly készítette:

Munkácsi Gergő, Varga Márk, Sinkó Kristóf

2023.09.26:

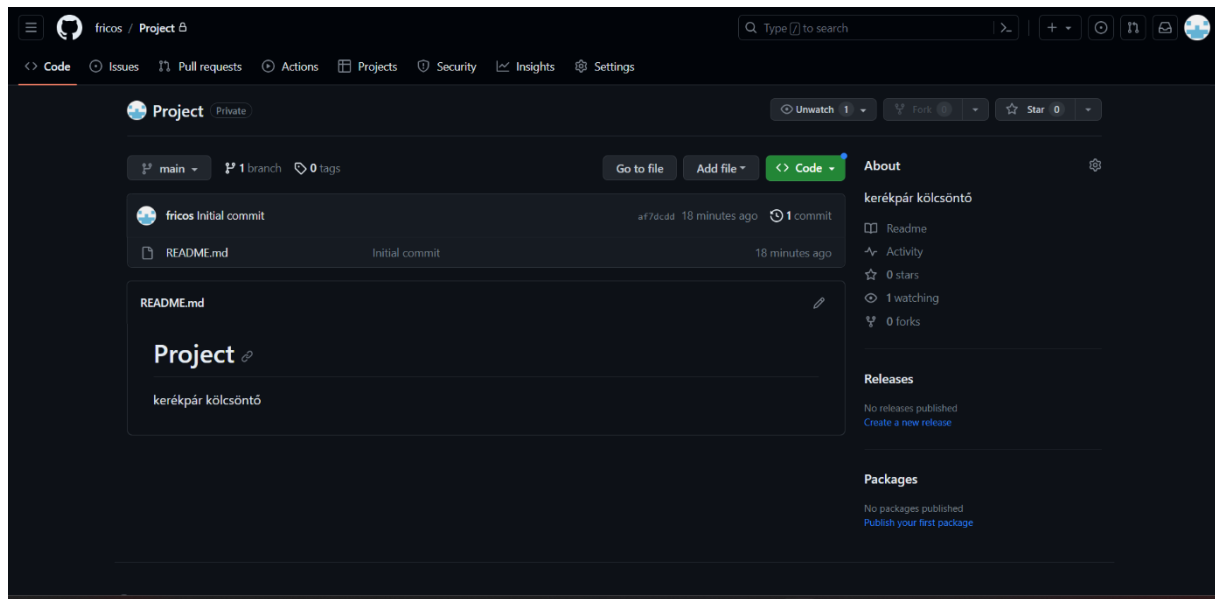
Ezen a napon a csoport összeült, hogy megbeszéljük mit szeretnénk csinálni, így tartottunk egy kisebb ötlet börzét.

Több mindent kigondoltunk, mint például egy olyan oldalt, amin az emberek értékelhetik az adott sorozatot/filmet és emellett írhatnak véleményt is róluk. A következő ötlet pedig egy olyan oldal lett volna, ami szintén egy filmes és sorozatos témában van.

Ezen az oldalon pedig mutatta volna a népszerű sorozatokat és filmeket plusz betudta volna állítani a kategóriát ezzel segítve, hogy mit nézzen.

A következő ötlet egy biciklikölcsönző volt. Ezen az oldalon pedig a felhasználó kibírná választani az adott biciklit, hogy elektromos vagy hagyományos legyen. Mindezek mellett megtudja adni a felhasználó, hogy mennyi időre szeretné kibérelni az adott.

Sok tanakodás és megbeszélés után, nagy nehezen döntöttünk, hogy melyik ötlet jön be nekünk végül pedig az utolsó, a biciklikölcsönző mellett maradtunk, mert az mindegyikünknek bejött és emellett, úgy gondoltuk hasznos is tud lenni, olyan embereknek, akik kirándulni vannak és nem tudják magukkal vinni a saját biciklijüket. Így a következő oldalakon a projektünk lépéseit fogjuk bemutatni, leírni és elmondani.



Először létrehoztunk a projektünknek egy repository-t, így a későbbiekben ide tudjuk majd feltölteni a projektünk egyes szakaszait. Ezáltal mindegyikünk tudni fogja, hogy hol tartunk/tartottunk legutoljára. Így a GitHub lesz az egyik olyan eszköz, ami segít nekünk a projekt végre hajtására. Ezt a weboldalt többnyire projektek megvalósításához és program kódok feltöltésére találták ki.

```
Munkacsi@MSI MINGW64 ~
$ git clone https://github.com/fricos/Project
Cloning into 'Project'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

Munkacsi@MSI MINGW64 ~
$ git pull
fatal: not a git repository (or any of the parent directories): .git

Munkacsi@MSI MINGW64 ~
$ git pull
fatal: not a git repository (or any of the parent directories): .git

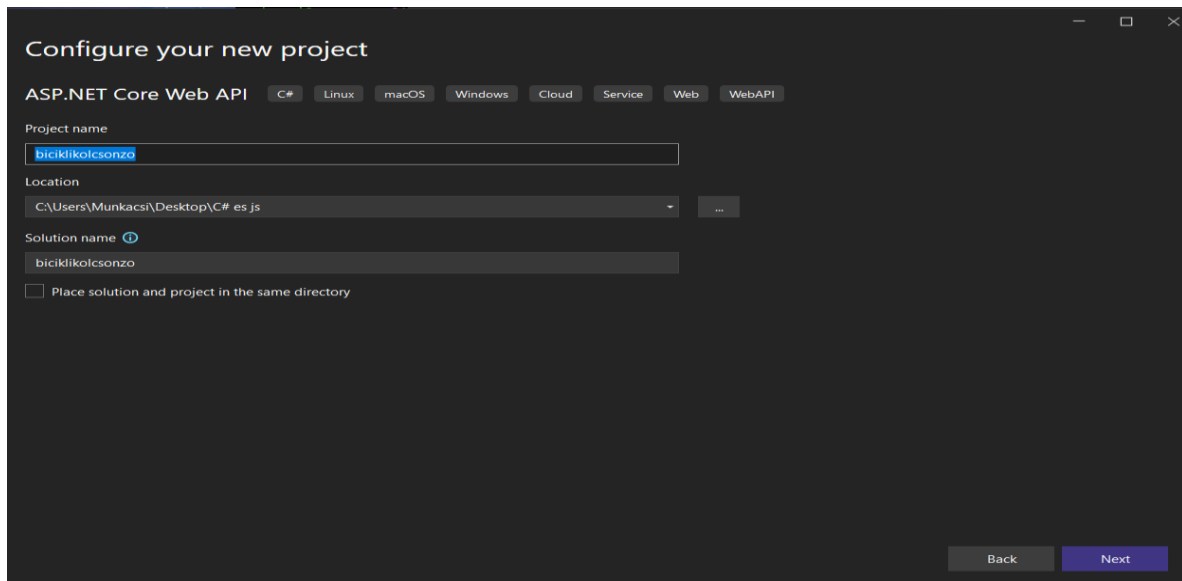
Munkacsi@MSI MINGW64 ~
$ git clone https://github.com/fricos/Project
fatal: destination path 'Project' already exists and is not an empty directory.

Munkacsi@MSI MINGW64 ~
$ cd Project

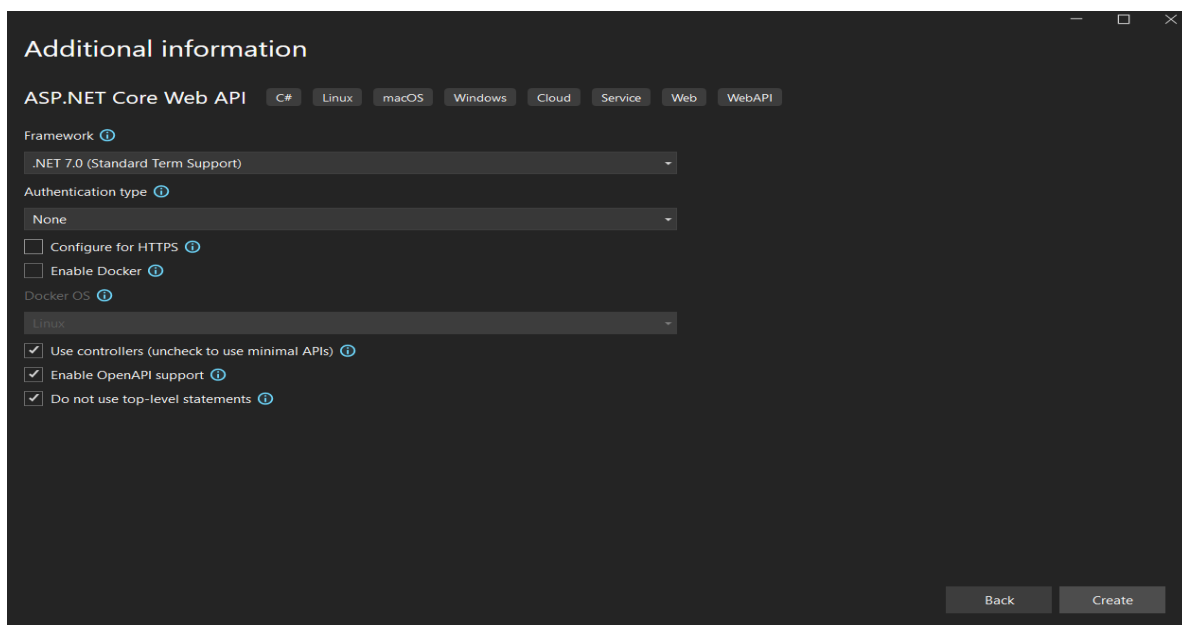
Munkacsi@MSI MINGW64 ~/Project (main)
$ git pull
Already up to date.

Munkacsi@MSI MINGW64 ~/Project (main)
$
```

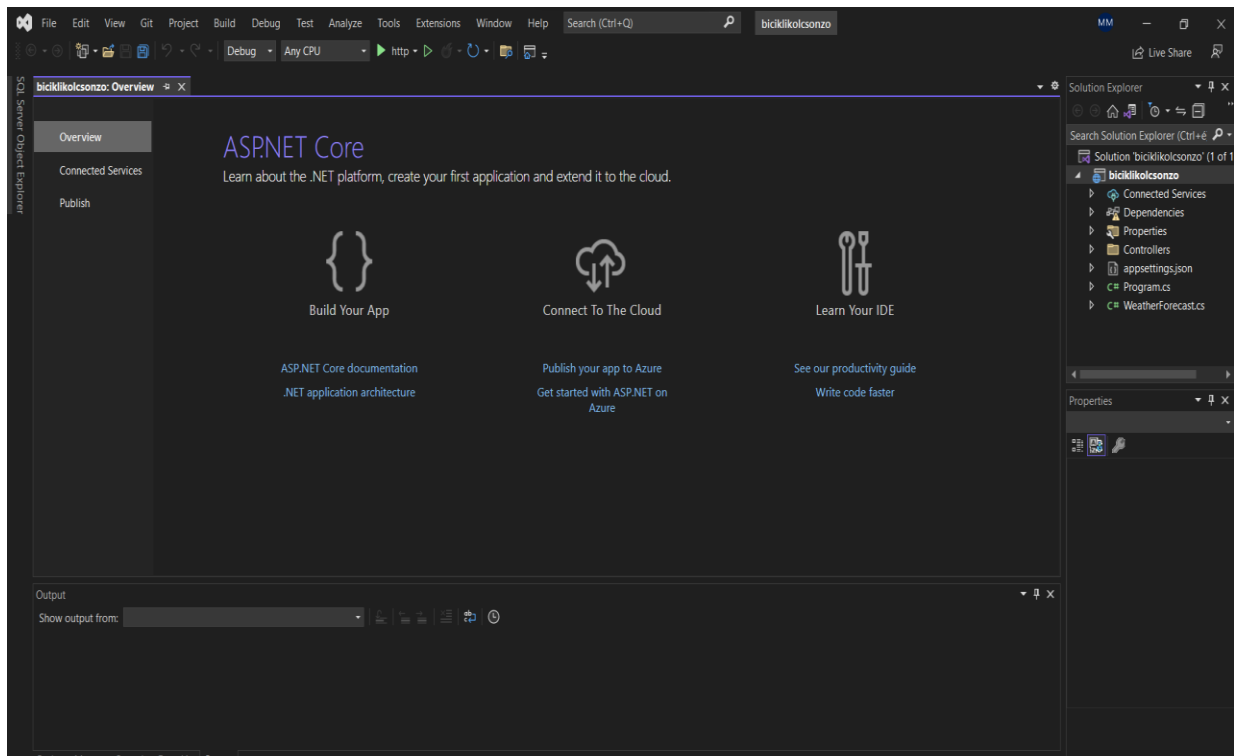
Itt a Git Bash látható segítségével az előző képen látható repositoryba tudunk fájlokat letölteni és emellett képes feltölteni is. A képen éppen a Git Hub-ról hívjuk le a repositorynkat aminek a neve Project, hogy lehívtuk a Projectet így már képesek vagyunk bele fájlokat rakni, ha raktunk bele fájlokat akkor azokat hasonló képpen feltudjuk tölteni.



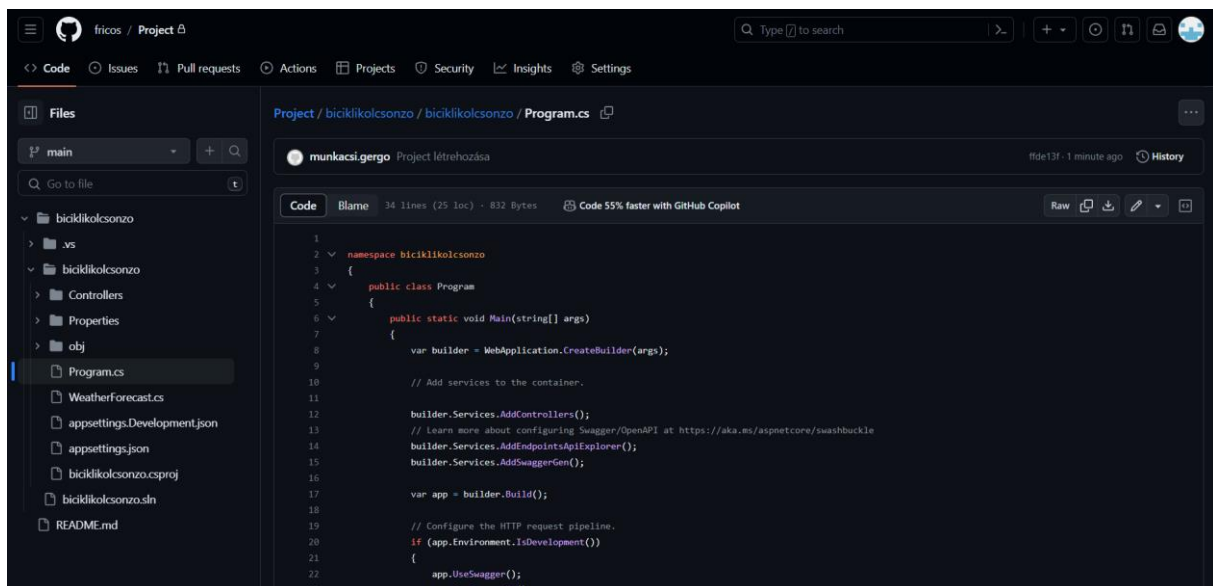
A két képen a Visual Studio 2022-ben létrehoztunk egy ASP.NET Core Web API-t, amit ezután elneveztünk.



Ezután pedig beállítottuk, úgy a konfigurációt ahogy mi, azt szeretnénk. Iyen például a Framework, aminek a .NET 7.0-t adtuk meg. Authentication type-nál kivettünk mindent. Ezután engedélyeztük, hogy használjon kontrollereket, támogassa az OpenAPI-t és ne használjon top-level statemneteket. De végül nem C# program nyelv mellett döntöttötünk hanem a Java mellett.



Ezután létrehoztuk a projektünket, ahol neki tudtunk állni a backend részének. Az utolsó képen az aznapi programunkat lehet látni. Ezt a programot a fent említett Git Bash-sel



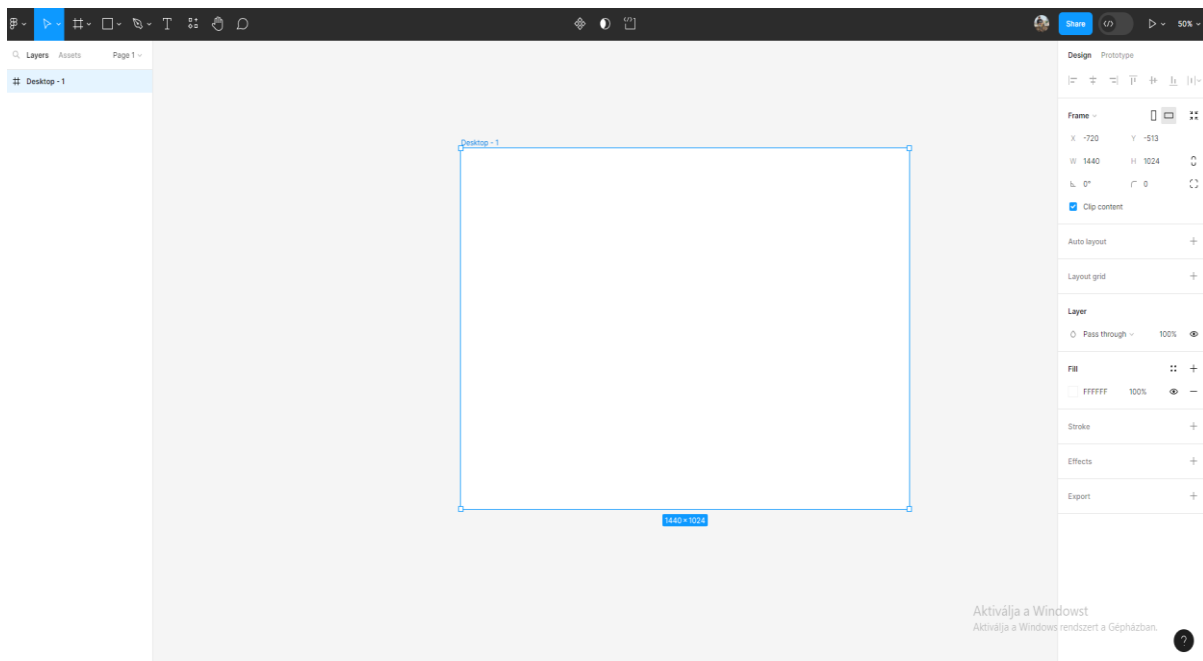
feltöltöttük a Project nevezetű repositorynkba.

2023.10.03:

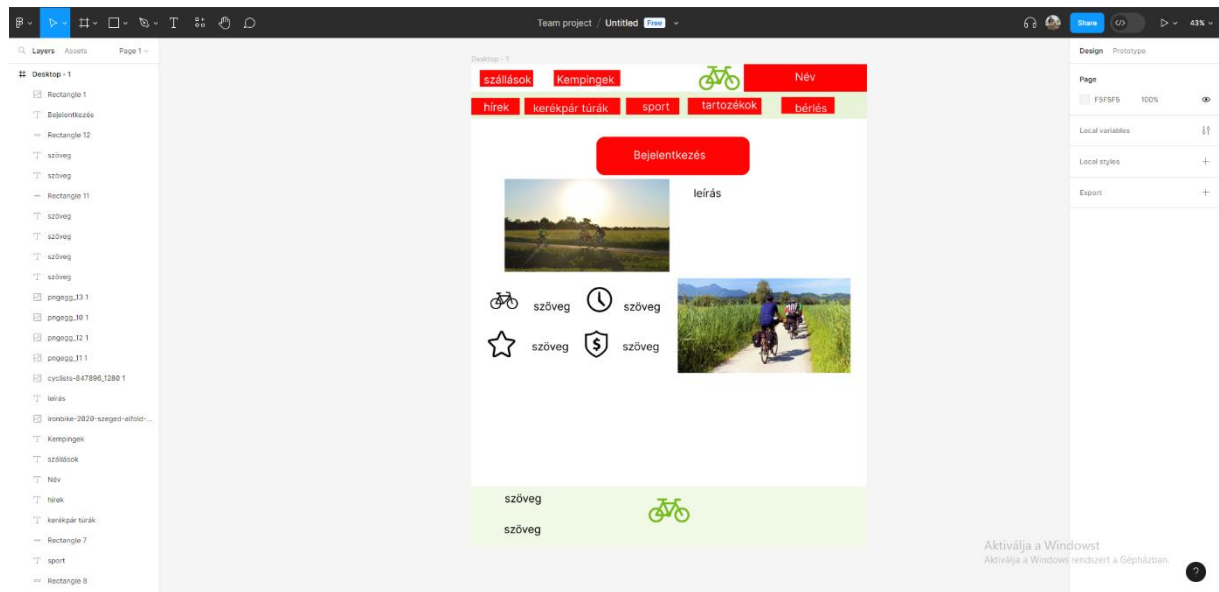
Ezen a napon a csoportunk összeült egy újabb megbeszélésre, ahol elkezdtünk arról beszélni, hogy miként nézzen ki az oldal, de mindezek előtt meg néztünk egy videót az ASP.NET Core-ról, hogy hogyan kell használni.

Ez egy kisebb időbe telt mire végig néztük. Mindezek után elkezdtünk képeket és ötleteket gyűjteni, hogy miként nézzen is ki a weboldalunk főoldala. Végül együtt elkezdtük a projektünk főoldalának vázlatát megcsinálni a Figma segítségével.

Az alábbi képen először létrehoztunk egy üres projektet.



Majd ezt elkezdtük feltölteni, képekkel és pár alakzattal, hogy mi hol legyen. Ezt a következő oldalon majd látni lehet, hogyan oldottuk meg.



A képen a projektünk főoldalának vázlata látszik. Felül az oldal jobb sarkába gondoltuk elhelyezni a weboldalunk nevét. Mellette balra az oldal logója látható. Az oldal baloldalán pedig elhelyeztünk két gombot az egyik a szállásokért fog felelni a másik pedig a kempingekért.

Ezek alatt található egy Nav bar. Erre még öt darab gombot ilyen például a: hírek, kerékpártúrák, sport, tartozékok és a bérlet.

Alatta található a bejelentkezés gomb, ahol a felhasználó tud majd jelentkezni az oldalra. Bejelentkezés alatt a kép mellett egy leírás lesz a túrákról. A másik kép mellett pedig az ikonoknál pedig, hogy mikor vagyunk nyitva, milyen biciklik vannak, fontos tudnivalók és az áraink.

Mindezek alatt pedig található, majd egy footer rész, ahol az elérhetőségünket, a helyszínt és a támogatóinkat lehet, majd megtalálni.

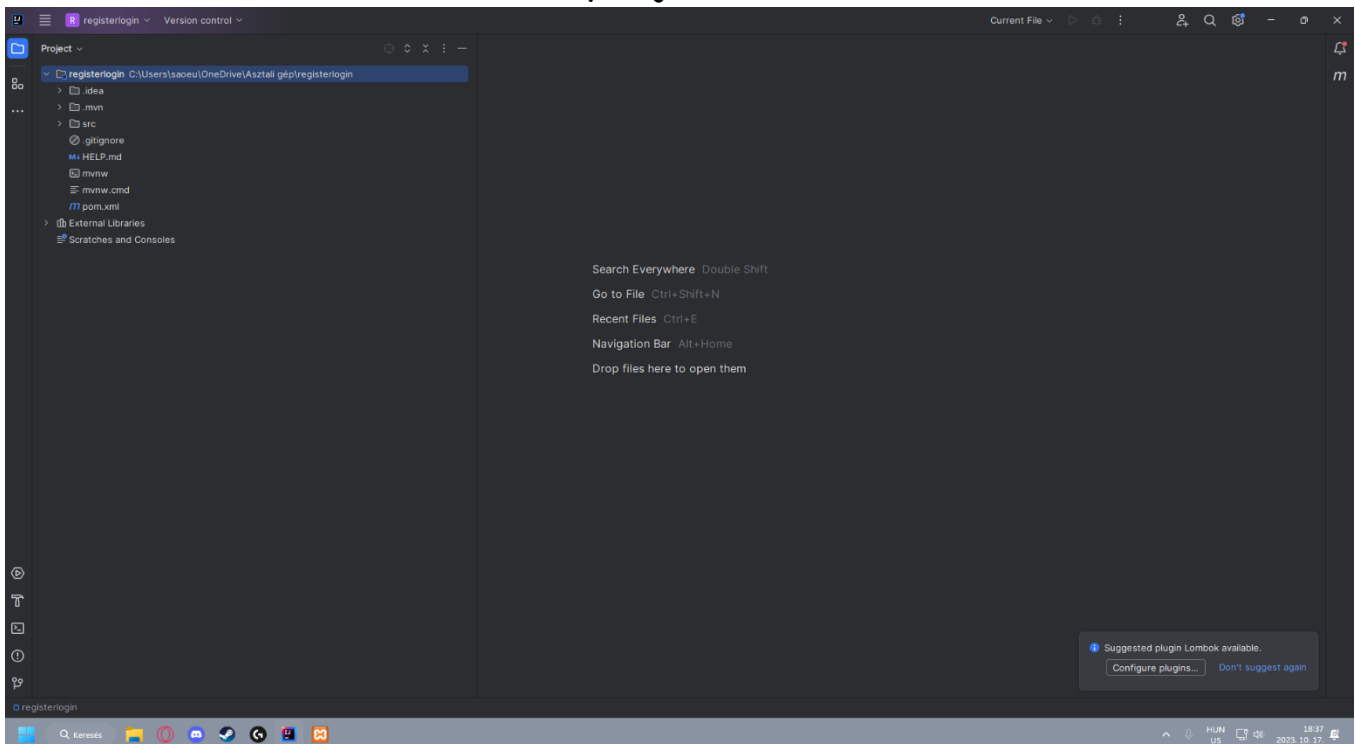
2023.10.17:

Ezen a napom a csapatunk összeült egy újabb megbeszélésre, hogy mit csináljunk következőnek, de végül arra jutottunk, hogy elkezdjük meg írni a programunkat. Ezt a java és a spring segítségével hoztunk létre egy regisztrációs és bejelentkezés programot mind ezt az IntelliJ IDEA segítségével. De mind ezek előtt beállítottuk a dependency-eket. Ezt az előbbiképen lehet látni.

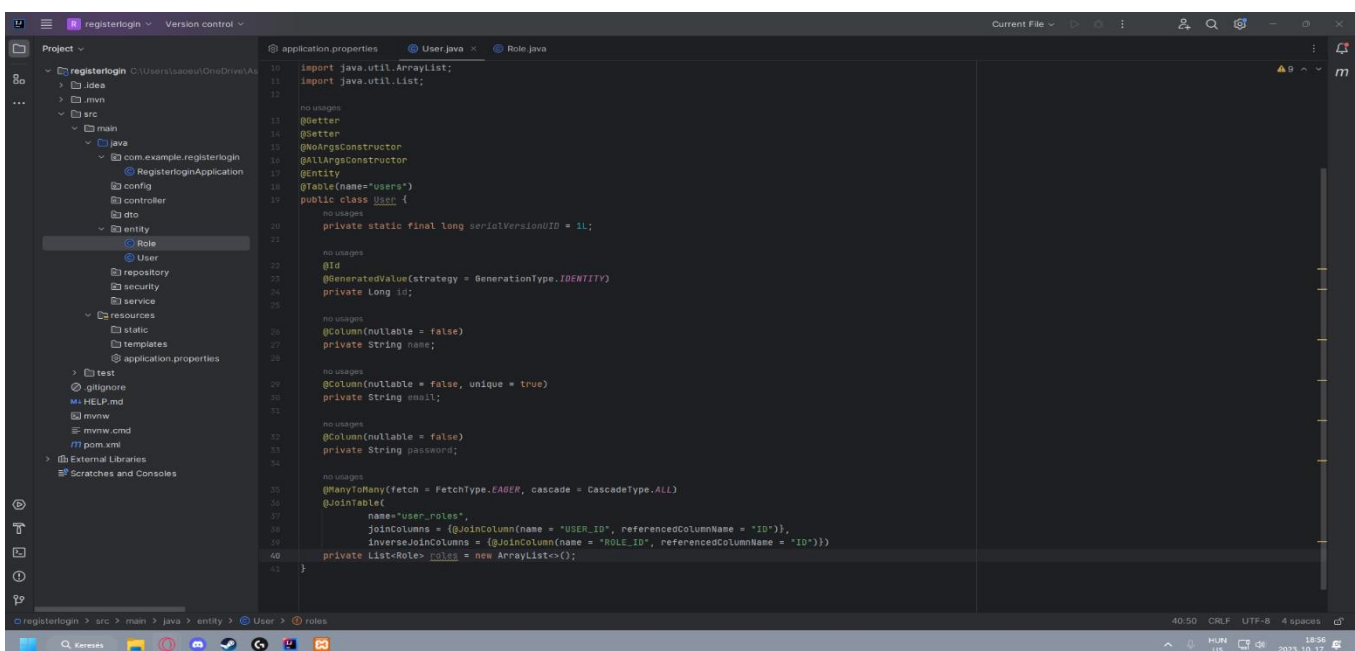


Itt beállítottuk, hogy milyen legyen a projekt, utána milyen nyelvben íródjon, melyik spring verziót használja, mi legyen a projekt metaadatai, ezután a packaginget állítottuk be majd, hogy milyen java verziót használjon. Mindezek végén hozzáadtuk, hogy miket használjon.

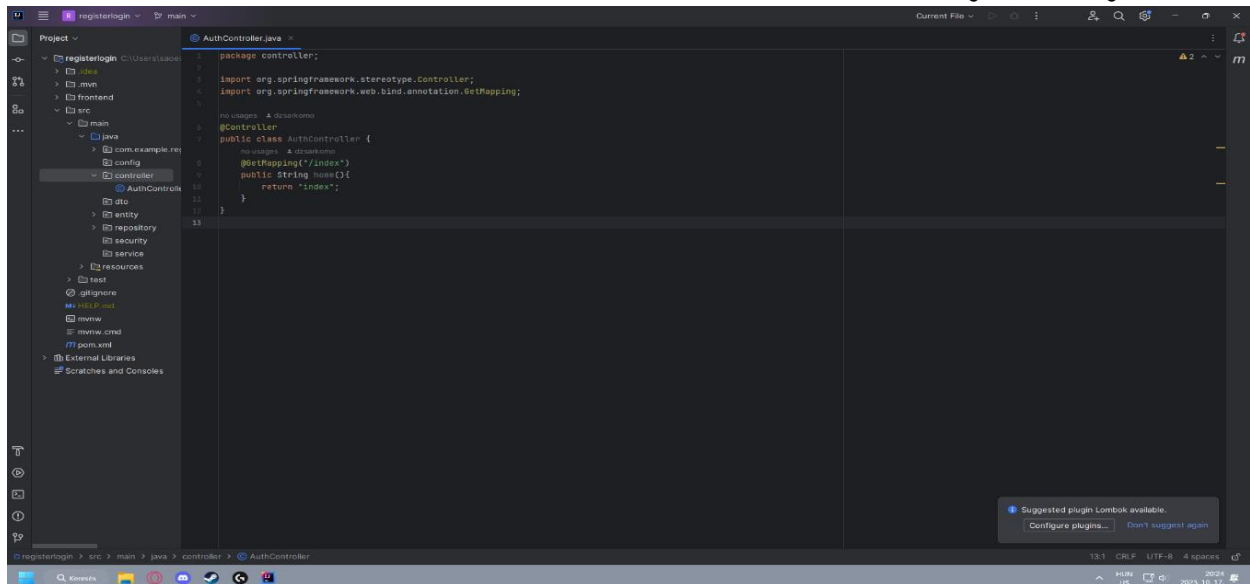
Ezek után létrehoztuk a projektet, amit itt lehet látni:



Miután ez megvolt az src mappán létrehoztuk a User.java-t és a Role.java fájlt és egy AuthController.java-t. A User.java fájl arra szolgál, hogy a felhasználókhöz több hozzáférést is lehet adni. Ennek a kód sora látható.



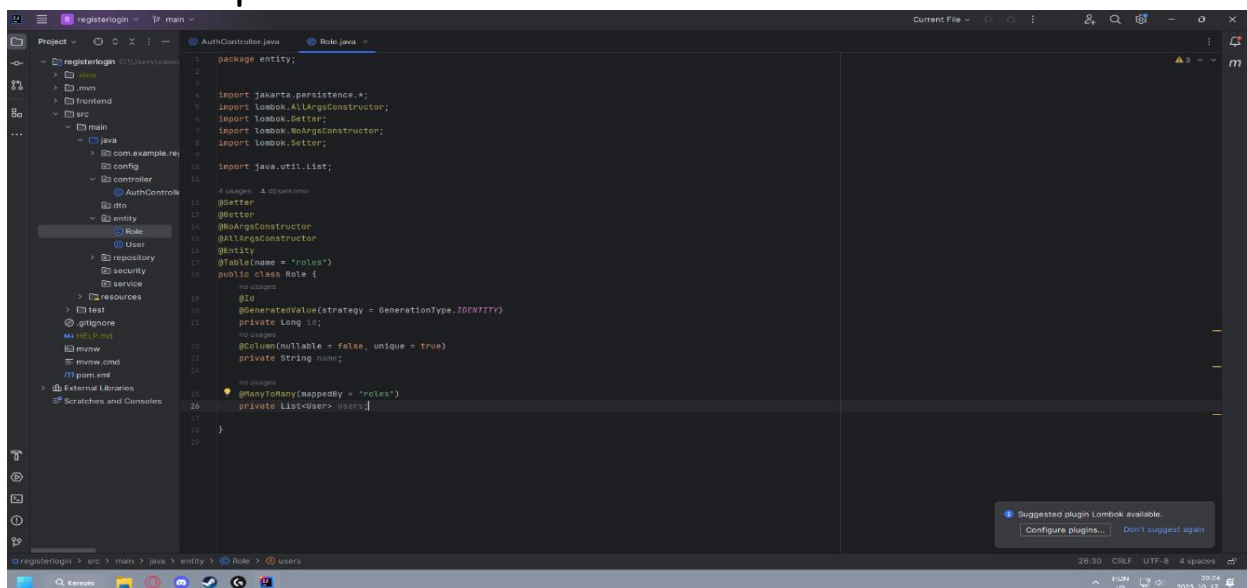
A kód megírása után nekiálltunk megírni az AuthController.java fájlban a kódját, ami arra szolgál, hogy el tudjon minket vinni a weboldalunk főoldalához ennek a kódrésze látható. Ezek után neki álltunk a Role.java fájlban a



megírásához.

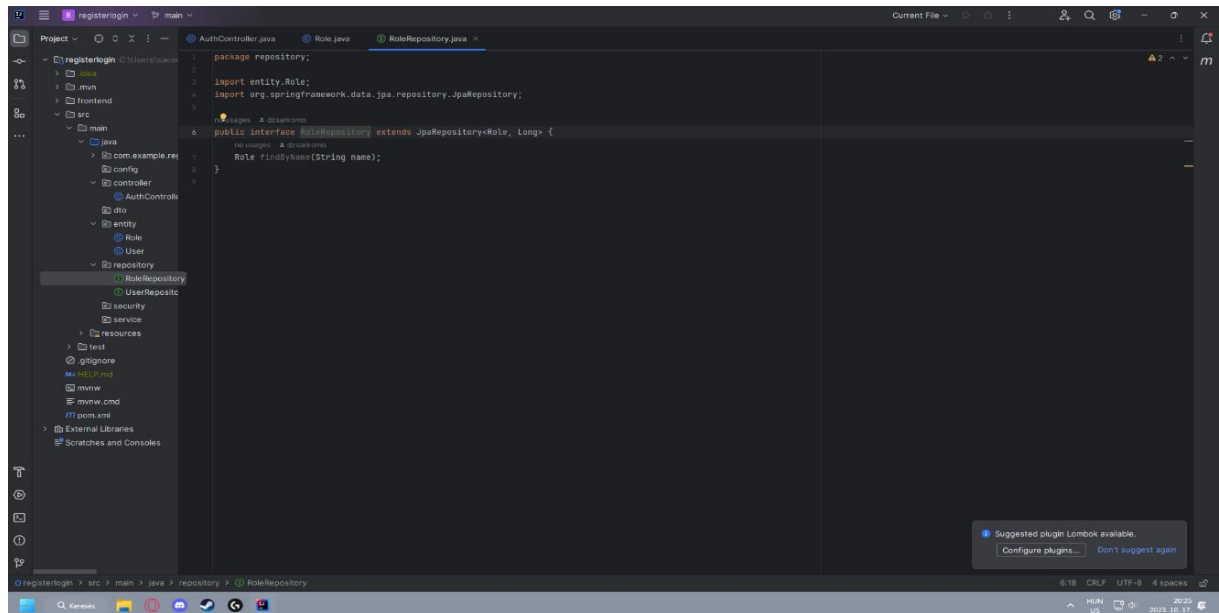
Ez a kódsor annyit tesz, hogy a felhasználónak oda tudjuk adni a szükséges hozzáféréseket, amik kellenek nekik.

A kód a képen látható.

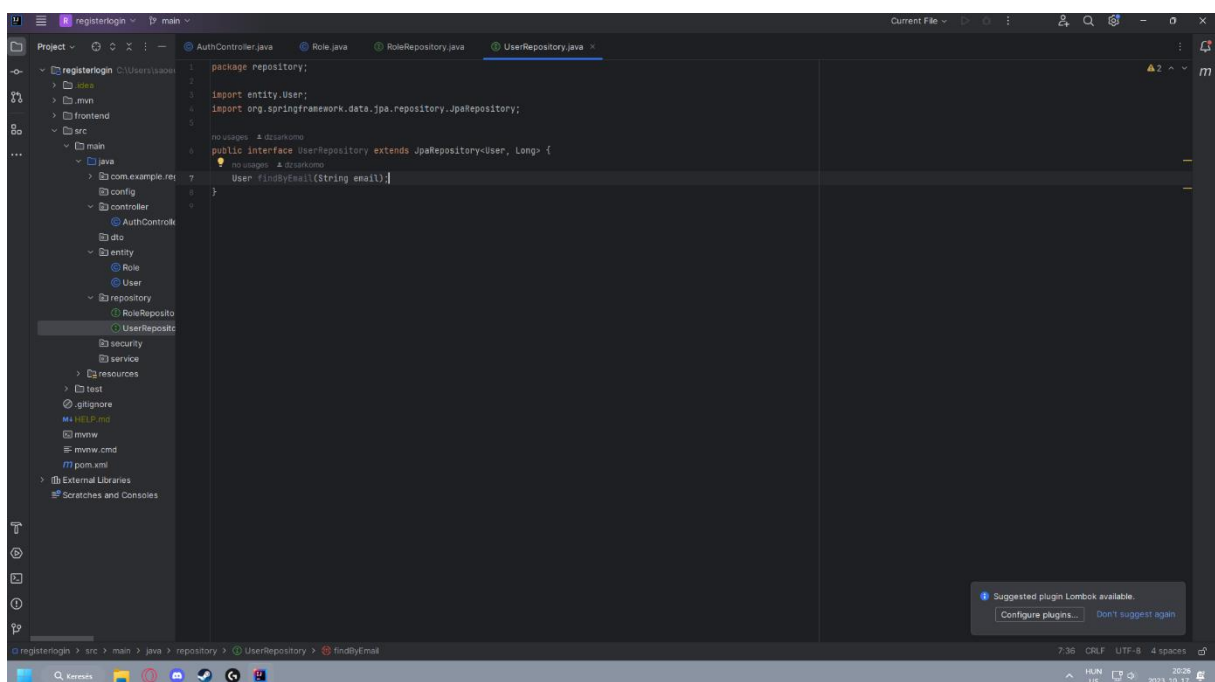


Ezek után a repository mappán létrehoztunk újabb fájlokat az egyik egy RoleRepository.java és egy UserRepository.java fájlt.

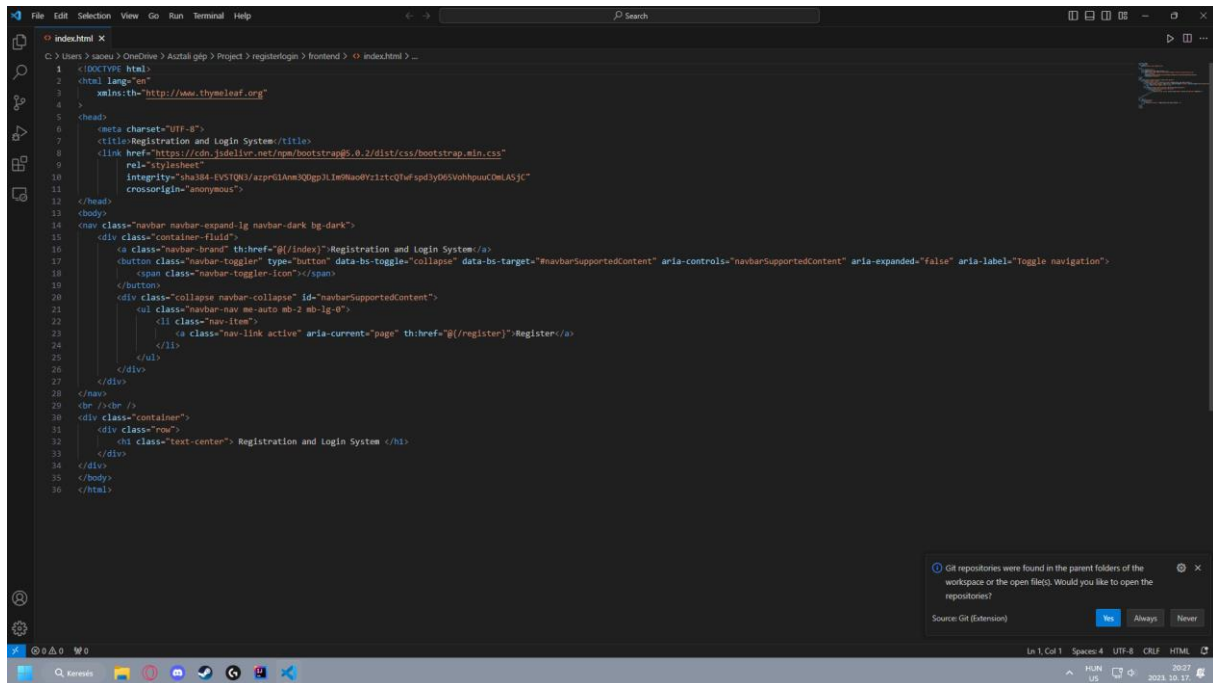
A kódos itt található.



Ennek a segítségével megtudjuk adni a felhasználónak milyen jogosultsága legyen. Az utóbbi a UserRepository.java kódja. Ez segít a regisztrációban és a bejelentkezésben is egyaránt.



Mindezek után meg írtuk a főoldal beléptető és regisztrációs felületét a Visual Studio Code-dal. Mindezt html, css valamint esetenként js használatával írtuk meg aminek a kódját itt lehet látni:

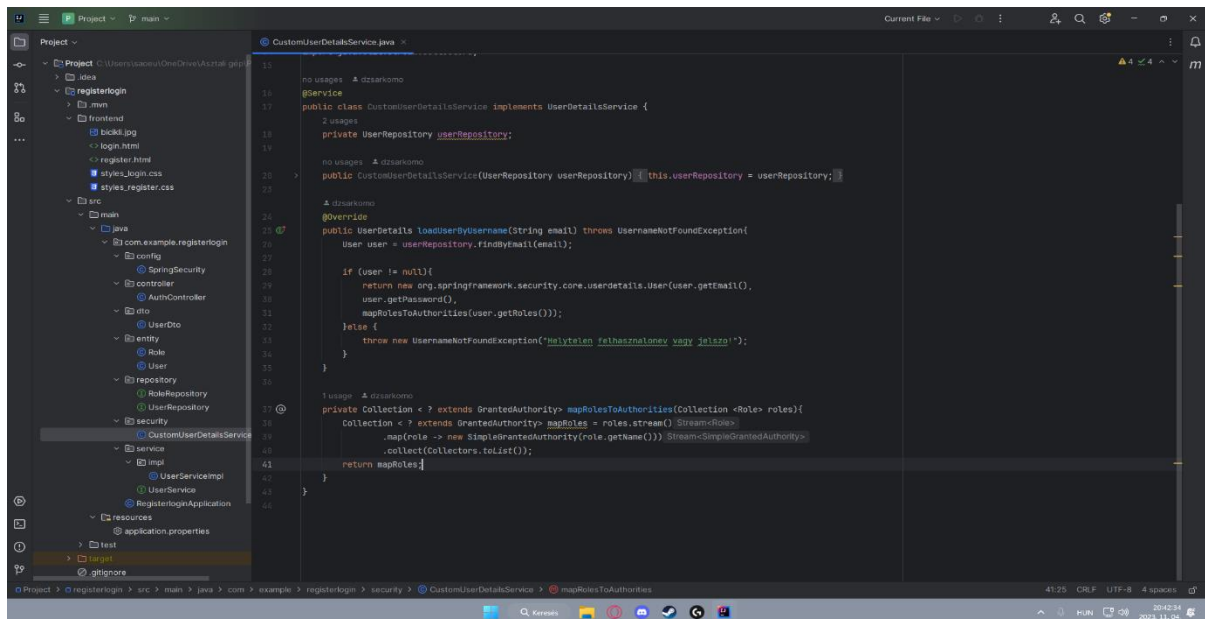


```
1 <!DOCTYPE html>
2 <html lang="en">
3   <meta charset="UTF-8">
4   <title>Registration and Login System</title>
5   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
6     rel="stylesheet"
7     integrity="sha384-E5YSTQh1/azp61Am33Qh33ImMaoRv11tcQh47spdy66Vhphus00L53j"
8     crossorigin="anonymous">
9   </head>
10  <body>
11    <div class="navbar navbar-expand-lg navbar-dark bg-dark">
12      <div class="container-fluid">
13        <a class="navbar-brand" href="#">Registration and Login System</a>
14        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent"
15          aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
16          <span class="navbar-toggler-icon"></span>
17        </button>
18        <div class="collapse navbar-collapse" id="navbarSupportedContent">
19          <ul class="navbar-nav me-auto mb-2 mb-lg-0">
20            <li class="nav-item">
21              <a class="nav-link active" href="#">Home</a>
22            </li>
23            <li class="nav-item">
24              <a class="nav-link active" href="#">Register</a>
25            </li>
26          </ul>
27        </div>
28      </div>
29    </div>
30    <div class="container">
31      <div class="row">
32        <div class="col">
33          <div class="text-center">Registration and Login System</div>
34        </div>
35      </div>
36    </div>
37  </body>
38 </html>
```

Ennek a segítségével látható lesz a weboldalunk felülete, ahova beszeretnénk, majd lépni.

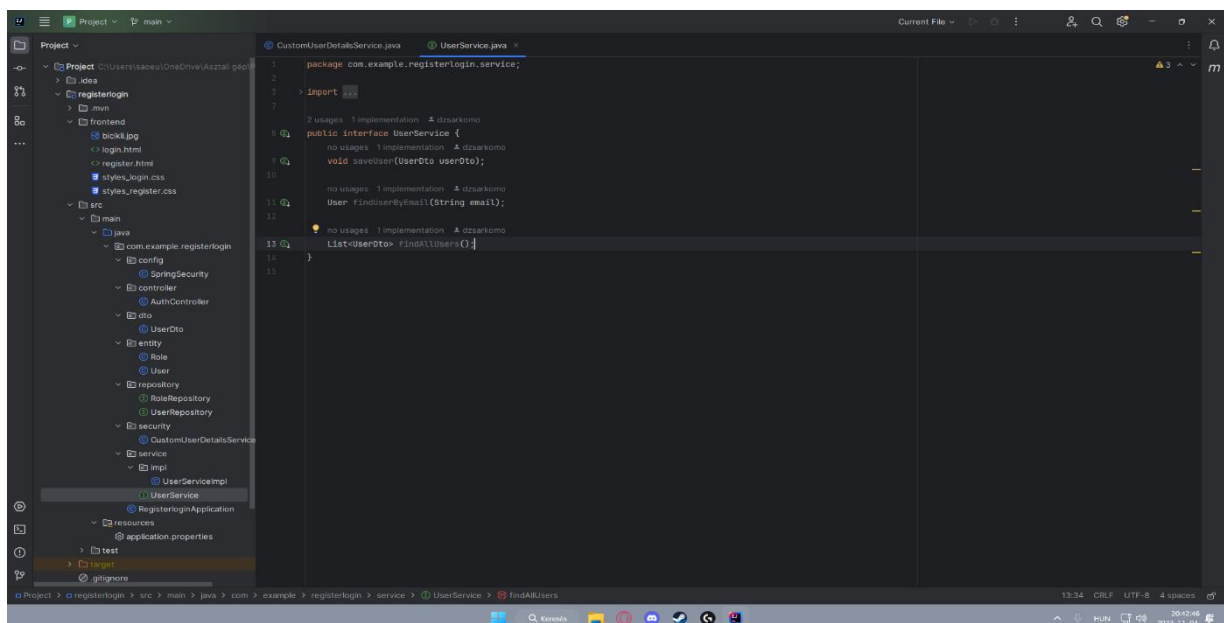
2023.11.04:

Ezen a napon csapatunk összeült egy újabb project munka megbeszéléshez. Ezen a napon folytattuk a Backendet. Kiegészítettük a Backendet egy egyedi felhasználó részlet szolgáltatással, aminek a kód sora itt található:



```
15
16 @Service
17 public class CustomUserDetailsService implements UserDetailsService {
18     private UserRepository userRepository;
19
20     public CustomUserDetailsService(UserRepository userRepository) { this.userRepository = userRepository; }
21
22     @Override
23     public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {
24         User user = userRepository.findByEmail(email);
25
26         if (user != null) {
27             return new org.springframework.security.core.userdetails.User(user.getEmail(),
28                 user.getPassword(),
29                 mapRolesToAuthorities(user.getRoles()));
30         } else {
31             throw new UsernameNotFoundException("Nem található felhasználó ezzel az email címmel");
32         }
33     }
34
35     private Collection<? extends GrantedAuthority> mapRolesToAuthorities(Collection<Role> roles) {
36         Collection<? extends GrantedAuthority> mapRoles = roles.stream().map(role -> new SimpleGrantedAuthority(role.getName()))
37             .collect(Collectors.toList());
38         return mapRoles;
39     }
40 }
```

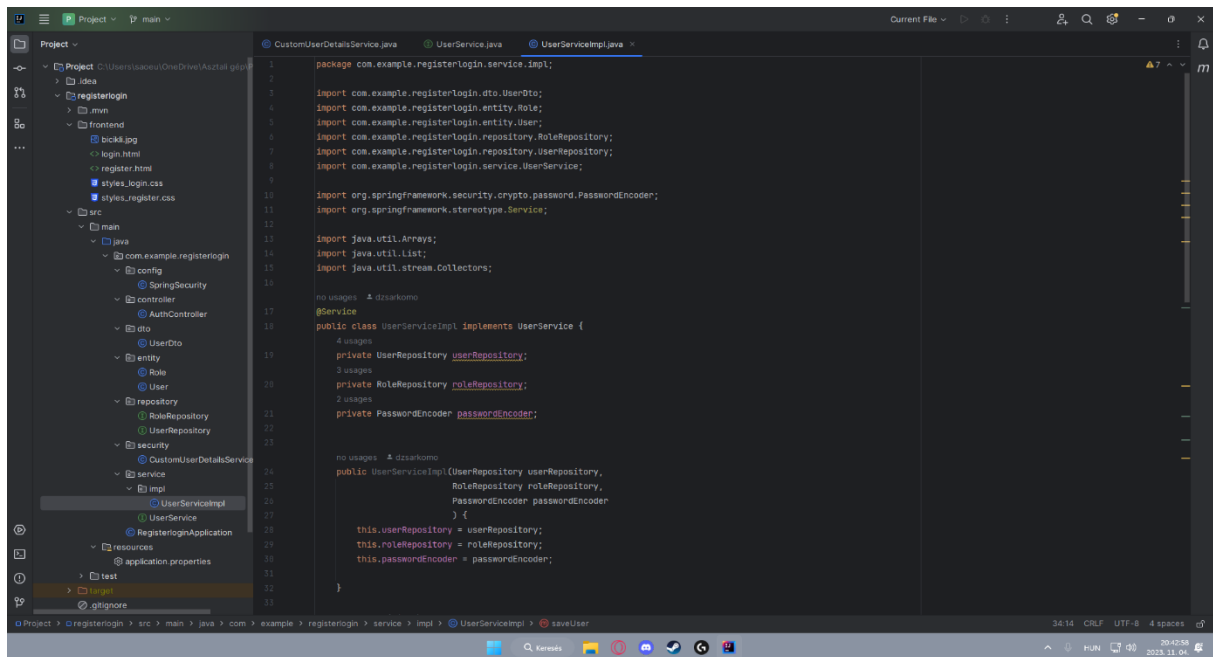
Ezután neki álltunk megcsinálni a felhasználó szolgáltatást, amit Márknak köszönhetően megcsináltunk. A program sort itt lehet látni:



```
1 package com.example.registerlogin.service;
2
3 import org.springframework.security.core.userdetails.UserDetails;
4
5 public interface UserService {
6     void saveUser(UserDto userDto);
7
8     UserDetails findUserByEmail(String email);
9
10    List<UserDto> findAllUsers();
11 }
```

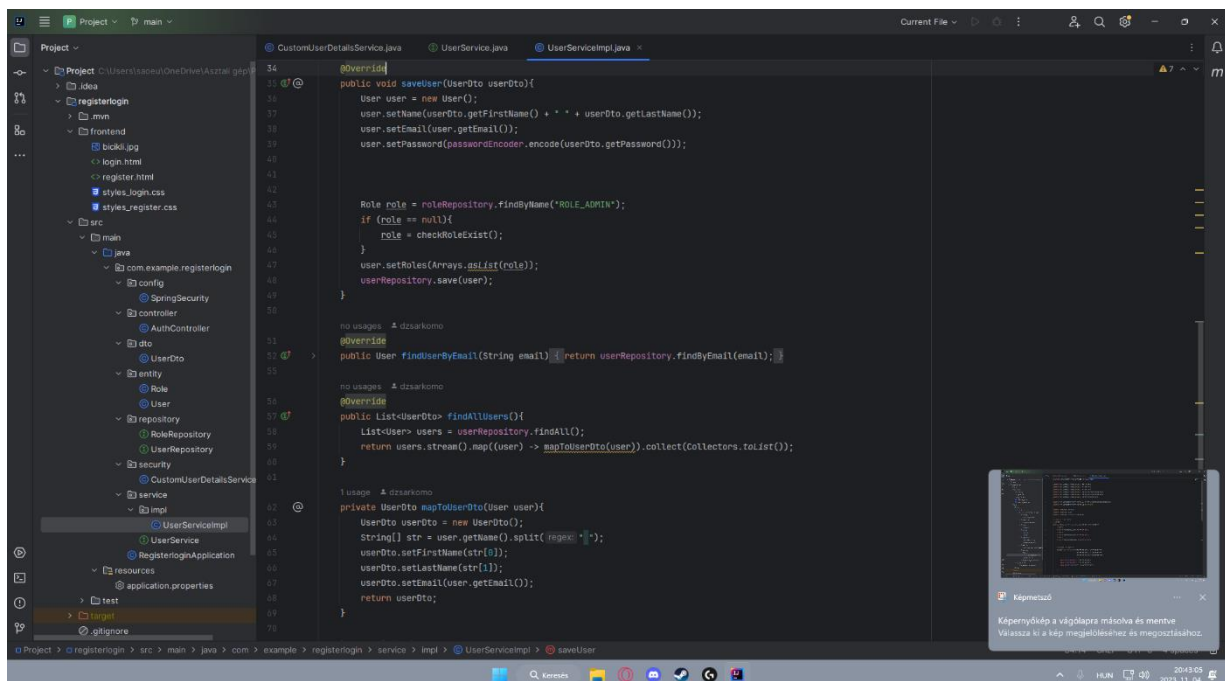
Mindezek után nagy nehezen ugyan, de sikerült meg írni a következő program sorunkat megírni, ami a felhasználó szolgáltatás implementációja volt. Ennek a kódsorai itt találhatóak meg:

A kód sor első rész



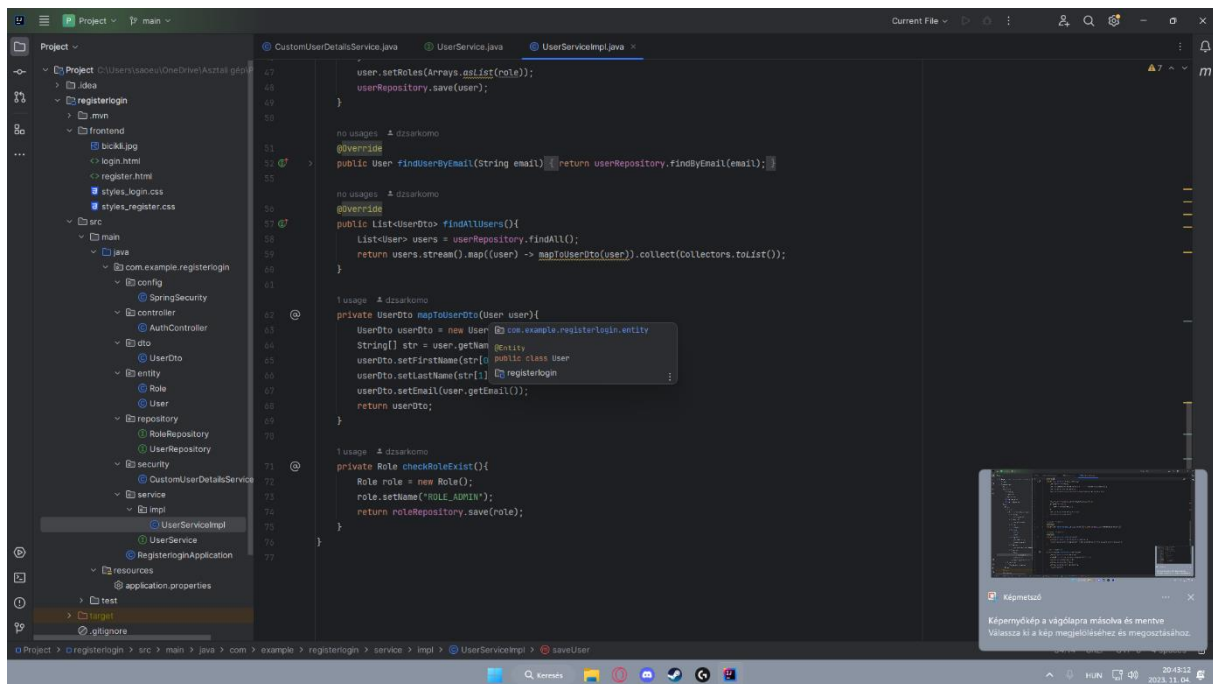
```
1 package com.example.registerlogin.service.impl;
2
3 import com.example.registerlogin.dto.UserDto;
4 import com.example.registerlogin.entity.Role;
5 import com.example.registerlogin.entity.User;
6 import com.example.registerlogin.repository.RoleRepository;
7 import com.example.registerlogin.repository.UserRepository;
8 import com.example.registerlogin.service.UserService;
9
10 import org.springframework.security.crypto.password.PasswordEncoder;
11 import org.springframework.stereotype.Service;
12
13 import java.util.Arrays;
14 import java.util.List;
15 import java.util.stream.Collectors;
16
17 no usages 1 d3zarkomo
18 @Service
19 public class UserServiceImpl implements UserService {
20
21     4 usages
22     private UserRepository userRepository;
23     3 usages
24     private RoleRepository roleRepository;
25     2 usages
26     private PasswordEncoder passwordEncoder;
27
28 no usages 1 d3zarkomo
29 public UserServiceImpl(UserRepository userRepository,
30     RoleRepository roleRepository,
31     PasswordEncoder passwordEncoder) {
32     {
33         this.userRepository = userRepository;
34         this.roleRepository = roleRepository;
35         this.passwordEncoder = passwordEncoder;
36     }
37
38 }
39
40 }
```

A második része



```
34 @Override
35 public void saveUser(UserDto userDto) {
36     User user = new User();
37     user.setName(userDto.getFirstName() + " " + userDto.getLastName());
38     user.setEmail(user.getEmail());
39     user.setPassword(passwordEncoder.encode(userDto.getPassword()));
40
41     Role role = roleRepository.findByName("ROLE_ADMIN");
42     if (role == null) {
43         role = checkRoleExist();
44     }
45     user.setRoles(Arrays.asList(role));
46     userRepository.save(user);
47 }
48
49 no usages 1 d3zarkomo
50 @Override
51 public User findUserByEmail(String email) {return userRepository.findByEmail(email); }
52
53 no usages 1 d3zarkomo
54 @Override
55 public List<UserDto> findUsers() {
56     List<User> users = userRepository.findAll();
57     return users.stream().map(user -> mapToUserDto(user)).collect(Collectors.toList());
58 }
59
60 1 usage 1 d3zarkomo
61 private UserDto mapToUserDto(User user) {
62     UserDto userDto = new UserDto();
63     String[] str = user.getName().split(" ");
64     userDto.setFirstName(str[0]);
65     userDto.setLastName(str[1]);
66     userDto.setEmail(user.getEmail());
67     return userDto;
68 }
69
70 }
```


Végül pedig ennek a kód sornak az utolsó része:



Mindezek után meg írtuk a felhasználói adatátviteli objektumot, itt megtudja adni a felhasználó az id-t, a keresztnévét, vezetéknévét, emailét és a jelszavát. Az utóbbi kettőnél megvan adva, hogy nem lehetnek üresek.

Program sor:

