Fejlesztési napló

Programrendszerek fejlesztése

Az alkalmazásom egy egyszerű webapplikáció, ahol történelmi pénzérméket lehet megtekinteni. Rendelkezik regisztrációval, bejelentkezéssel, egy főoldallal, egy érméket listázó oldallal és egy részletező oldallal. Kétféle felhasználó létezik, a normál és az admin. Az alapértelmezett admin neve „admin”, jelszava „admin123”.

A mondogDB-t Atlas segítségével hosztolom, a connection string ’mongodb+srv://dzseta:progrendfejlmongo@progrendfejl.xsmazas.mongodb.net/?retryWrites=true&w=majority', de ez szerepel a kódban. A csatlakozás minden IP-ről engedélyezve lett.

A program futtása: „npm install” parancs kiadása a szerver mappájában. Majd „node index.js” parancs. Ezek után localhost:3000-en található meg az applikáció.

Backend

A backend statikusan hostolja a frontendet a public mappából. (index.js)

```
app.use(express.static(path.join(__dirname, 'public')));
```

Az alkalmazás kapcsolódik egy mongodb instance-hoz Atlas segítségével. (index.js)

```
const dbUrl = 'mongodb+srv://dzseta:progrendfejlmongo@progrendfejl.xsmazas.mongodb.net/?retryWrites=true&w=majority';
mongoose.connect(dbUrl);
```

Az alkalmazás képes bootstrappelni, vagyis MongoDB-t alap userekkel feltölteni. Minden indításkor ellenőrzi, hogy létezik-e admin felhasználó, amennyiben nem, akkor generál egy újat. (bootstrap.js)

A szerver megvalósít legalább két modellt, melyek sémája egyértelműen definiált. A 2 modell a coinSchema.js és userSchema.js fájlokban található meg.

```
const userSchema = new mongoose.Schema({
    // a séma legfontosabb elemei az eltárolt dokumentumok adattagjai
  username: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
  },
  password: {
    type: String,
    required: true,
  },
  admin: {
    type: Number,
    required: true,
    default: 0,
  },
});
```

Adott legalább két olyan adatbázis hook, amelyek a modellek mentése vagy lekérése közben futnak le. (userSchema.js, routes.js)

```
userSchema.pre('save', function(next) {
  const user = this;
  if(user.isModified('password')) {
      user.admin = 0;
      bcrypt.genSalt(10, function(err, salt) {
          if(err) {
              console.log('hiba a salt generalasa soran');
              return next(error);
          }
          bcrypt.hash(user.password, salt, function(error, hash) {
              if(error) {
                  console.log('hiba a hasheles soran');
                  return next(error);
              }
              user.password = hash;
              return next();
          })
      })
  } else {
      return next();
  }
});
```

A szerver megvalósít egy lokális autentikációs stratégiát passport segítségével. (index.js)

A szerver kezeli a login sessiont. (routes.js)

A szerver rendelkezik a két kezelt modell CRUD interfészeivel, illetve egy login, logout, register route-tal. (routes.js)

```
// login
passport.use('local', new localStrategy(function (username, password, done) {
    User.findOne({ username: username }, function (err, user) {
        if (err) return done('Hiba lekeres soran', null);
        if (!user) return done('Nincs ilyen felhasználónév', null);
        user.comparePasswords(password, function (error, isMatch) {
            if (error) return done(error, false);
            if (!isMatch) return done('Hibas jelszo', false);
            return done(null, user);
        })
    })
}));
```

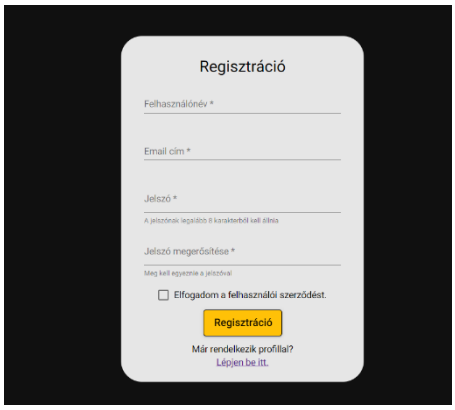
Frontend

A frontend kommunikál a backenddel.

```
export const environment = {
  production: true,
  url: 'http://localhost:3000/'
};
```

A frontend komponensei route-okkal érhetőek el, a navigáció megfelelően működik. (app_routing_module.js)

A frontend rendelkezik legalább egy regisztráció, egy login, egy főoldal/terméklista, egy admin felület, egy termék részletező és egy egyéb komponenssel, melyek fel vannak töltve megfelelő tartalommal. (login, register, coins, admin, coindetails, error)



A frontend a bejelentkezéshez a backend megfelelő végpontjait szólítja meg. (*.service.ts)

A backenddel való kommunikáció elemei ki vannak szervezve service-ekbe.

(*.service.ts)

Van authguard, amely védi a login, register utáni route-okat és az admin felületét. (authguard.guard.ts)

```
export class AuthguardGuard implements CanActivate {

  constructor(private router: Router) {
    }
  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
```