웹 어플리케이션 보안
- QRCode와 Captcha 활용

조영선

91613935

사용된 기술

1. JWT(토큰 인증 로그인)
2. QR Code
3. reCAPTCHA

웹 어플리케이션 진행 과정

01

Mongo DB와 Nodejs에 서비스 구상

02

Angular로 Fornt-End 구상

03

클라우드 서비스 이용

Mongo DB & Node.js

UserSchema 생성

```
const UserSchema = mongoose.Schema({
  name: {
    type: String
  },
  email: {
    type: String,
    required: true
  },
  major: {
    type: String,
    required: true
  },
  hakbun: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  }
});
```

Mongo DB & Node.js

DB에서 정보 찾기

```
const User = module.exports = mongoose.model('User',
UserSchema);

module.exports.getUserById = function(id, callback){
  User.findById(id, callback);
}

module.exports.getUserByHakbun = function(hakbun, callback){
  const query = {hakbun: hakbun};
  User.findOne(query, callback);
}
```

User라는 이름으로 외부에서 사용할수 있도록 모델 생성하고 Export

DB에서 id로 사용자 확인하는 함수 생성
함수를 외부에서 사용할 수 있도록 Export

DB에서 hakbun으로 사용자 검색

Mongo DB & Node.js

사용자 등록 기능 추가

```
module.exports.addUser = function(newUser, callback){
  bcrypt.genSalt(10, (err, salt) => {
    bcrypt.hash(newUser.password, salt, (err, hash) => {
      if(err) throw err;
      newUser.password = hash;
      newUser.save(callback);
    });
  });
}
```

Password 암호화

DB에 저장

Mongo DB & Node.js

JWT(토큰 인증 로그인)

사용자 등록 기능 추가

```
router.post('/register', (req, res, next) => {

  let newUser = new User({
    name: req.body.name,
    email: req.body.email,
    major: req.body.major,
    hakbun: req.body.hakbun,
    password: req.body.password
  });

  User.addUser(newUser, (err, user)=>{
    if(err) {
      res.json({success: false, msg: '사용자 등록에 실패했습니다.'});
    } else {
      res.json({success: true, msg: '사용자 등록에 성공했습니다.'});
    }
  });
});
```

사용자가 입력창에서 입력하는 값들을 newUser 변수에 저장

Mongo DB & Node.js

JWT(토큰 인증 로그인)

패스포트 로그인

```
module.exports = function(passport){
  let opts = {};
  opts.jwtFromRequest = ExtractJwt.fromAuthHeaderWithScheme('jwt');
  opts.secretOrKey = config.secret;
  passport.use(new JwtStrategy(opts, (jwt_payload, done) => {
    User.getUserById(jwt_payload.data._id, (err, user) => {
      if (err) {
        return done(err, false);
      }
      if (user) {
        return done(null, user);
      } else {
        return done(null, false);
      }
    });
  }));
}
```

opts
- 토큰을 전송
- 토큰 생성을 위한 비밀키 설정

패스포트에 JWT strategy 적용

App.js 에서 passport.js를 읽어옴

```
require('./config/passport')(passport);
```

Mongo DB & Node.js

JWT(토큰 인증 로그인)

인증 기능 추가

```
router.post('/authenticate', (req, res, next) => {
  const hakbun = req.body.hakbun;
  const password = req.body.password;

  User.getUserByHakbun(hakbun, (err, user) => {
    if(err) throw err;
    if(!user) {
      return res.json({success: false, msg: '사용자 정보가 없습니다.'});
    }

    User.comparePassword(password, user.password, (err, isMatch) => {
      if(err) throw err;
      if(isMatch) {
        const token = jwt.sign({data:user}, config.secret, {
          expiresIn: 604800  // 1 week
        });
```

사용자가 입력하는 로그인 정보를 변수로 할당

hakbun으로 user를 DB에서 검색

입력 패스워드와 DB에서 읽어온 패스워드 정보를 비교

패스워드가 일치하면 토큰을 생성 유효기간 설정

Mongo DB & Node.js

JWT(토큰 인증 로그인)

인증 기능 추가

```
res.json({
        success: true,
        token: 'JWT '+token,
        user: {
          id: user._id,
          name: user.name,
          email: user.email,
          major: user.major,
          hakbun: user.hakbun
        }
      });
    } else {
      return res.json({success: false, msg: '비밀번호를 확인해 주세요'});
    }
  });
});
});
```

토큰과 사용자 정보를 브라우저에 리턴

Mongo DB & Node.js

JWT(토큰 인증 로그인)

특정 루트 보호

```
router.get('/profile', passport.authenticate('jwt', {session:false}), (req, res, next) =>
{
  res.json({user:{
  id: req.user._id,
  name: req.user.name,
  hakbun: req.user.hakbun,
  major: req.user.major,
  email: req.user.email}
  });
```

토큰인증 된 경우에만 서버가
응답한 user 데이터를 보여줌

Angular

Fornt-end (QR Code, reCAPTCHA)

사용자 등록

```
export class RegisterComponent implements OnInit {
  name: string;
  email: string;
  major: string;
  hakbun: string;
  password: string;
  captchaResponse: string;
  captchavalue: string;

  }
```

사용자 등록 정보는 4개의 사용자 입력 정보로 사용

Recaptcha사용을 위한 값

Angular

Fornt-end (QR Code, reCAPTCHA)

사용자 등록

```
constructor(
  private validateService: ValidateService,
  private flashMessage: FlashMessagesService,
  private authService: AuthService,
  private router: Router  ) { }
```

서비스 등록

Angular

Fornt-end (QR Code, reCAPTCHA)

사용자 등록 폼

```
<h2 class="page-header">Register</h2>
<form (submit)="onRegisterSubmit()">
<div class="form-group">
<label> 이 름 </label>
<input type="text" [(ngModel)]="name" name="name" class="form-control"> </div>
```

Submit 버튼을 누르면 onRegisterSubmit()함수가 실행됨

2-way data binding

Angular

Fornt-end (QR Code, reCAPTCHA)

onRegisterSubmit() 함수

```
 onRegisterSubmit(){
const user = {
      name: this.name,
      email: this.email,
      major: this.major,
     hakbun: this.hakbun,
     password: this.password    }
 if(!this.validateService.validateRegister(user)){
     this.flashMessage.show('모든 필드를 입력해주세요!', {cssClass: 'alert-danger', timeout: 3000});
     return false;    }
if(!this.validateService.validateEmail(user.email)){
     this.flashMessage.show('유효한 이메일을 입력해주세요!', {cssClass: 'alert-danger', timeout: 3000});
      return false;    }
```

사용자가 폼에 입력한 값

모든 필드를 입력하지 않을 시 뜨는 flashMessage

유효하지 않은 이메일을 입력 시 뜨는 flashMessage

Angular

Fornt-end (QR Code, reCAPTCHA)

onRegisterSubmit() 함수

사용자가 폼에 입력한 값을 Authservice의 registerUser로 전달

```
this.authService.registerUser(user).subscribe(data => {
if(data.success) {
this.flashMessage.show('사용자 등록 성공, 로그인 하세요!', {cssClass: 'alert-success', timeout: 3000});
this.router.navigate(['/login']);     }
else {
    this.flashMessage.show('사용자 등록 실패', {cssClass: 'alert-danger', timeout: 3000});
    this.router.navigate(['/register']);     }   }); }
```

사용자 등록 성공시 login화면으로
실패시 register 화면으로 이동

Angular

Fornt-end (QR Code, reCAPTCHA)

registerUser()함수

```
registerUser(user){
  let headers = new Headers();
  headers.append('Content-Type', 'application/json');
  return this.http.post('users/register', user, {headers: headers}).pipe(map(res => res.json()));  }
```

사용자 정보 서버로 전달

Angular

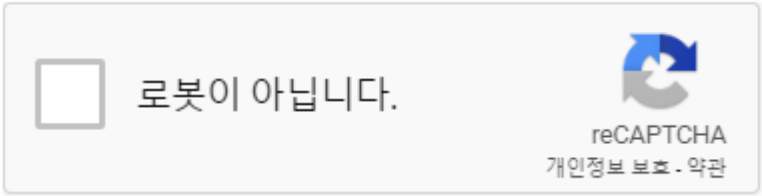
Fornt-end (QR Code, reCAPTCHA)

사용자 등록

상자 체크시 이벤트 발생

```
<re-captcha (resolved)="resolved($event)"
siteKey="6LdwuIEUAAAAAPGIhm1Kbwu_yBXnJ9QZ3dN9PWH8"></re-captcha> <br>
```

구글에서 제공하는 reCAPTCHA 사이트에서 받은 키 값



reCAPTCHA 띄우기

```
<input *ngIf="response()" type="submit" class="btn btn-success" value="Submit"></form>
```

reCHATPCHA인증이 완료되면 sumbit버튼 띄우기

SUBMIT

Angular

Fornt-end (QR Code, reCAPTCHA)

Resolved()함수와 response() 함수

```
resolved(captchaResponse) {
  this.captchavalue = `${captchaResponse}`; }
```

이벤트 발생시 생성 되는 값

```
response(){   return this.captchavalue; }
```

Captchavalue가 존재할 경우
Sumbit버튼 띄우기

Angular

Fornt-end (QR Code, reCAPTCHA)

Login 폼

```
<h2 class="page-header">Login </h2>
<form (submit)="onLoginSubmit()">
  <div class="form-group">
    <label>학 번</label>
    <input type="text" class="form-control" [(ngModel)]="hakbun" name="hakbun">
  </div>
  <div class="form-group">
    <label>비 밀 번 호</label>
    <input type="password" class="form-control" [(ngModel)]="password" name="password">
  </div>
  <input type="submit" class="btn btn-success" value="Login"></form>
```

Login 버튼 클릭시
onLoginSubmit() 함수 실행

2-way databinding 설정

Angular

Fornt-end (QR Code, reCAPTCHA)

onLoginSubmit()

```
onLoginSubmit(){
  const user = {
    hakbun: this.hakbun,
    password: this.password    }
this.authService.authenticateUser(user).subscribe(data => {
    if(data.success){
      this.authService.storeUserData(data.token, data.user);
      this.flashMessage.show('로그인 되었습니다!', {cssClass:'alert-success', timeout:5000});
     this.router.navigate(['profile']);      }
else {
      this.flashMessage.show(data.msg, {cssClass: 'alert-danger', timeout:5000});
      this.router.navigate(['login']);      }    });  }
```

Hakbun과 password가 존재하면
로그인되며 profile 페이지로 이동

토큰과 user데이터
localStorage에 저장

Hakbun과 password가 존재하지 않
으면 login 페이지로 이동

Angular

Fornt-end (QR Code, reCAPTCHA)

authenticateUser()함수

```
authenticateUser(user){
    let headers = new Headers();
    headers.append('Content-Type', 'application/json');
    return this.http.post('users/authenticate', user, {headers: headers}).pipe(map(res => res.json()));  }
```

사용자 정보 서버로 전달

storeUserData()함수

```
storeUserData(token, user){
    localStorage.setItem('id_token', token);
    localStorage.setItem('user', JSON.stringify(user));
    this.authToken = token;
    this.user = user;  }
```

로그인 성공시
-토큰과 user데이터를 로컬스토리지에 저장
- 토큰, user데이터를 this.authToken과 this.user 변수로 저장

Angular

Fornt-end (QR Code, reCAPTCHA)

logout()함수

```
logout(){
  this.authToken = null;
  this.user = null;
  localStorage.clear();  }
```

로그인 변수의 값을 지우고
로컬스토리지도 지운다

로그아웃 버튼

로그인 된경우에 띄우기

```
<li *ngIf="authService.loggedIn()" class="nav-item" >
<a class="nav-link" style="color: #FFFFFF" (click)="onLogoutClick()" href="#">로그아웃</a></li>
```

Angular

Fornt-end (QR Code, reCAPTCHA)

onLogoutClick()함수

```
onLogoutClick(){
    this.authService.logout();
    this.flashMessage.show('로그아웃 되었습니다!', {cssClass: 'alert-success', timeout: 3000});
    this.router.navigate(['/login']);
    return false;  }
```

Angular

Fornt-end (QR Code, reCAPTCHA)

getProfile() 함수

```
getProfile(){
    let headers = new Headers();
    this.authToken = localStorage.getItem('id_token');
    headers.append('Authorization', this.authToken);
    headers.append('Content-Type', 'application/json');
    return this.http.get('users/profile', {headers: headers}).pipe(map(res => res.json()));  }
```

토큰을 읽어와서 authorization 헤더에 첨부

ngOnInit() 함수 → 자동으로 실행되게 하고싶을 때 사용하는 함수

```
ngOnInit() {
    this.authService.getProfile().subscribe(profile => {
      this.user = profile.user;
    }, err => {
    console.log(err);
    return false;  });  }
```

서버에서 제공하는 사용자 정보 적용

Angular

Fornt-end (QR Code, reCAPTCHA)

프로필 페이지

로그인 된 경우에만 사용자 정보 표시

```
<div *ngIf="user">
  <h2 class="page-header">{{user.name}}</h2>
  <ul class="list-group">
    <li class="list-group-item">학과 : {{user.major}}</li>
    <li class="list-group-item">학번 : {{user.hakbun}}</li>
    <li class="list-group-item">이메일 : {{user.email}}</li>
  </ul></div>
```

Angular

Fornt-end (QR Code, reCAPTCHA)

QR Code 페이지

로그인 된 경우에만 사용자 정보 표시

```
<div *ngIf = user>
 <div id="container">
  <div id="block">
   <h2 class="page-header">QR 코드 데이터</h2>
    <ul class="list-group">
      <h4>{{user.name}}님의 QR코드 </h4>
     <li class="list-group-item">학과 : {{user.major}}</li>
     <li class="list-group-item">학번 : {{user.hakbun}}</li>
     <li class="list-group-item">이메일 : {{user.email}}</li>
    </ul>
    <br>
    <button (click) = "createCode()" class="btn btn-success " > QR 코드 생성하기 </button>      <br>
    <div *ngIf="createdCode">
     <ngx-qrcode [qrc-element-type]="img" [qrc-value] = "createdCode"></ngx-qrcode>      <div>
 </div>
</div></div>
```

버튼을 누를시 createCode() 함수 실행

QR Code가 생성된 경우에만 표시

Angular

Fornt-end (QR Code, reCAPTCHA)

QR Code ts파일

```
Suser : string;
 user = Object;
 createdCode = null;
```

```
ngOnInit() {
   this.authService.getProfile().subscribe(profile => {
     this.user = profile.user;
   }, err => {
   console.log(err);
   return false;  });  }
```

프로필 정보 자동으로 가져오기

```
createCode(){
   this.authService.getProfile().subscribe(profile => {
     this.Suser = JSON.stringify(profile.user);
     this.createdCode = this.Suser;
   }, err => {
   console.log(err);    return false;  });}
```

프로필 정보를 이용해서 QR Code 생성

QR Code를 만들기 위해 JSON -> String

Angular

Fornt-end (QR Code, reCAPTCHA)

생성된 QR Code

QR 코드 데이터

김민경님의 QR코드

학과 : 정보보호

학번 : 91613624

이메일 : alsrud@naver.com

QR 코드 생성하기

QR 코드 데이터

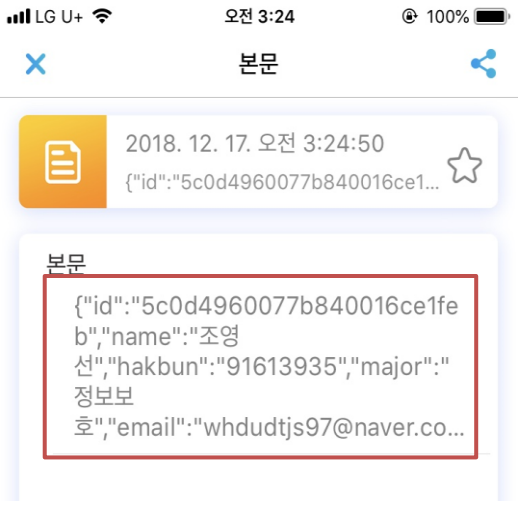
조영선님의 QR코드

학과 : 정보보호

학번 : 91613935

이메일 : whdudtjs97@naver.com

QR 코드 생성하기



Angular

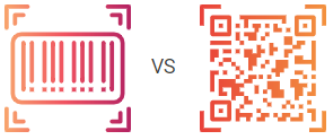
Fornt-end (QR Code, reCAPTCHA)

QR Code 와 reCAPTCHA 페이지

QR 코드란 ?

바코드보다 훨씬 많은 정보를 담을 수 있는 격자 무늬의 2차원 코드이다.
스마트폰으로 QR [Quick Response] 코드를 스캔하면 각종 정보를 제공받을 수 있다.

QR code 장점



기존 바코등 비해 많은 정보를 담으면서 코드 크기는 바코드에 비해 작다.

정사각형의 모양의 큐알코드는 360어느방향에서든 인식가능

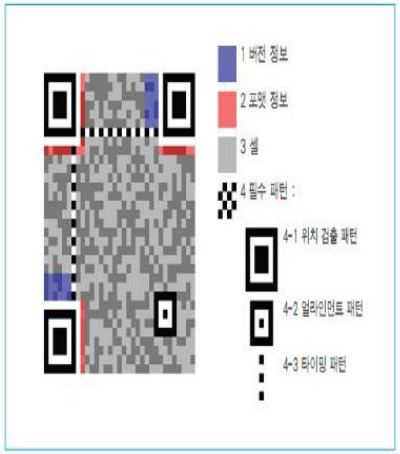
QR code의 구조

Angular

Fornt-end (QR Code, reCAPTCHA)

QR Code 와 reCAPTCHA 페이지

QR code의 구조



QR코드의 구조

출처 | Zephyris (영문 위키백과)

캡챠 (CAPTCHA) 란 ?

사람과 완전 자동화된 컴퓨터(혹은 로봇 또는 봇)을 판별하기 위한 일종의 디지털 보안 프로그램 기술이라고 할 수 있습니다.

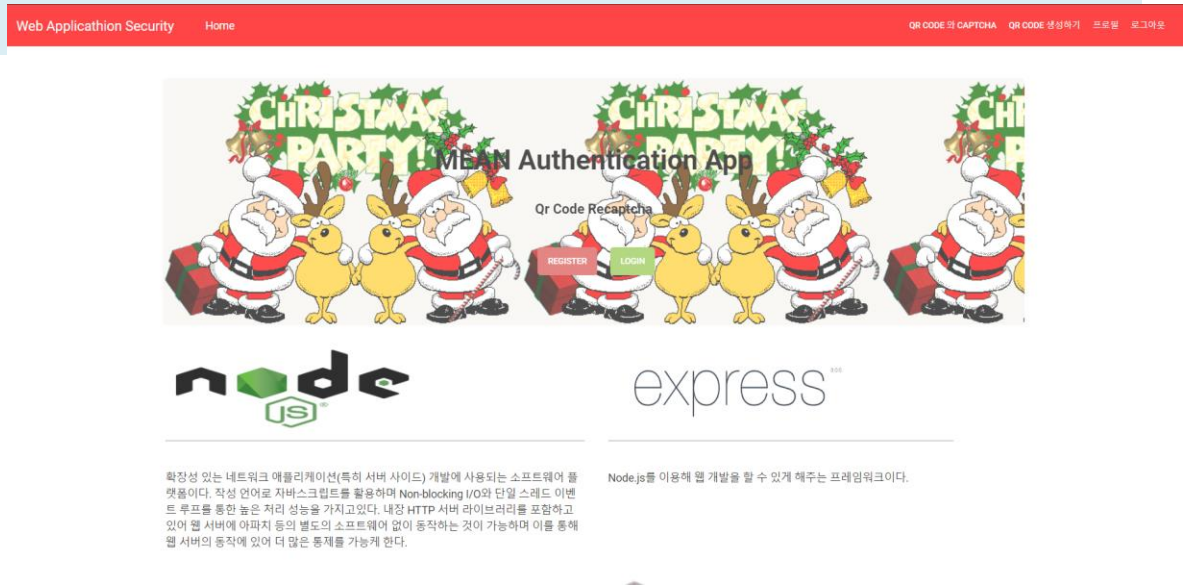
CAPTCHA는 Completely Automated Public Turing test to tell Computers and Humans Apart의 약어이다.

Angular

Fornt-end (QR Code, reCAPTCHA)

홈 페이지 화면

```
<div class="jumbotron text-center image">
  <h1>MEAN Authentication App</h1>
  <p class="lead">Qr Code Recaptcha</p>
  <div>
    <a class="btn btn-danger" [routerLink]="['/register']">Register</a>
    <a class="btn btn-success" [routerLink]="['/login']">Login</a>
  </div>
</div>
```



클라우드 서버 사용

Heroku와 mlabDB 사용

```
database: 'mongodb://youngsun:youngsun123@ds015720.mlab.com:15720/yy'
```

<DB user> : <DB 비밀번호>

mlabDB에 등록한 DB 불러오기

```
const port = process.env.PORT || 3000;
```

Heroku 서버의 port를 불러오기 위해
Process.env.PORT 사용

감사합니다!