Express: Passport.js

Питонофилы могут использовать для выполнения задания Authomatic. https://github.com/authomatic/

Passport.js

- Passport — это middleware для проверки подлинности
- Passport – это коллекция готовых плагинов для различных провайдеров
- После использования Passport волосы становятся мягкими и шелковистыми.

Терминология

- Стратегия (Strategy) – поведенческий паттерн проектирования.
- Стратегия – реализация взаимозаменяемых алгоритмов, инкапсулирующих детали реализации
- Стратегия Passport.js – отдельный NPM-пакет passportjs.org/packages 500+ стратегий
- Можно использовать готовые, можно создавать свои, не забывая потом вносить вклад в OpenSource

Стратегии

- passport-local
- passport-vk-strategy passport-vkontakte
- passport-yandex
- passport-github
- passport-ldap passport-activedirectory
- passport-something-else

Passport Init

```
const session = require("express-session"),
    bodyParser = require("body-parser");
const passport = require("passport");


app.use(session({ secret: "supersecret" }));
app.use(bodyParser.urlencoded({ extended: false }));
app.use(passport.initialize());
app.use(passport.session());


//passportjs.org/docs
```

Local Strategy

```
const LocalStrategy = require("passport-local").Strategy;

passport.use(new LocalStrategy((username, password, done)=>{
        findUser(username,(err, user)=>{
            if (err) {
                return done(err);
            }
            if (!user) {
                return done(null, false);
            }
            if (password !== user.password ) {
                return done(null, false);
            }
            return done(null, user);
        })
    }
));
```

findUser

- Поиск пользователя в базе данных приложения любыми способами
    - Mongo
    - MySQL
    - Postgree
    - LDAP…

Сессии. Сериализация

```
passport.serializeUser(function(user, done) {
  done(null, user.id);
});
//user - объект, который Passport создает в req.user
```

```
passport.deserializeUser(function(id, done) {
  findUserById(id, function(err, user) {
    done(err, user);
  });
});
//Для минимизации можно хранить только ID пользователя.
```

Login

```
app.post('/login',
  passport.authenticate('local', {successRedirect: '/',
                                  failureRedirect: '/login'});
);
```

```
app.get(`/login', (req, res) => {
    res.render(some-login-form);
}
```

```
//passport.authenticate внутри вызывает req.login()
```

Is Auth? Middleware

```
const isAuth = (req, res, next) => {
      if (req.isAuthenticated()) {
          return next();
      }
      res.redirect('/');
      }

app.get('/', isAuth, (req, res) => {res.render(some-page);}
```

Logout

```
app.get('/auth/logout', isAuth, (req, res) => {
    req.logout();
    res.render(bye-bye-page); //or redirect
}
```

Разные провайдеры

```
app.get('/auth/provider',passport.authenticate('provider'));

app.get('/auth/provider/callback',
  passport.authenticate('provider',
                        { successRedirect: '/',
                          failureRedirect: '/login' }
                        ));
```

```
<a href="/auth/provider">Log In with OAuth Provider</a>
```

Исследуем VK

`passport-vk-strategy`

```
passport.use(new VKontakteStrategy(
   {
     clientID: VKONTAKTE_APP_ID,
     clientSecret: VKONTAKTE_APP_SECRET,
     callbackURL:  URL
  },
  (accessToken, refreshToken, params, profile, done)=>{
     …
  });
```