WebToDo++

Bogdan-Ștefan Cernat, Adrian Gheorghe Schipor

Facultatea de Informatică, Universitatea Alexandru Ioan Cuza, Iași
bogdan.cernat@info.uaic.ro

adrian.schipor@info.uaic.ro

Abstract

WebToDo++ este o aplicație web ce oferă utilizatorilor posibilitatea de a-și organiza sarcinile și proiectele. Utilizatorii pot crea proiecte și pot adăuga sarcini. În cazul sarcinilor ce beneficiază de un memento, serverul va notifica utilizatorul la ora și data respctivă.

Introducere

WebToDo++ ajută utilizatorii să își administreze timpul și sarcinile eficient oferind o cale de a-și organiza activitățile, de a seta priorități și memento-uri pentru sarcini. Avantajul aplicației constă in faptul că poate fi accesată oricând atât timp cât există o conexiune la internet. Aplicația are două moduri de utilizare: cei care nu dețin cont de utilizator pot doar să adauge sarcini și vor fi notificați doar atunci când vor accesa aplicația; cei care dețin cont de utilizator vor fi notificați prin email sau prin mesaj direct pe contul de *Twitter* și pot crea proiecte. Astfel, cei care vor să beneficieze de toate facilitățile oferite de această aplicație trebuie să își creeze un cont de utilizator. Deasemenea, aceștia se pot autentifica și folosind contul de Twitter, Google.

Tehnologii folosite

Partea de server este implementată modular folosind limbajul de programare **JavaScript** și rulează pe platforma **nodeJS**. *nodeJS* este o platformă *open source* care ușurează dezvoltarea de aplicații web scalabile în limbajul *JavaScript*. Această platformă adoptă paradigma bazată pe eventimente, operațiile de intrare fiind asincrone. Pentru a facilita dezvoltarea aplicației a fost folosit framework-ul *Express* impreună cu anumite module. Pentru salvarea datelor a fost folosită baza de date *NoSQL* **CouchDB**. Această bază de date stochează datele in documente *JSON* facilitând dezvoltarea de aplicații web. Deasemenea, este potrivită pentru crearea de aplicații web scalabile și sigure. Front-end-ul aplicației este realizat folosind *template engine-ul* **Jade**, template folosit pentru genererarea de conținut *HTML*. Pentru formatarea elementelor documentelor *HTML* este folosit *Stylus*. Comunicarea dintre utilizatori și

server este realizată folosind modulul *io.sockets*. *io.sockets* este un modul ce oferă suport *cross-browser* pentru comunicarea asincronă dintre client și server, și se poate folosi de *WebSockets*, *Ajax*, *Adobe® Flash® Socket* împreună cu alte modalități de comunicare asincronă, în funcție de capabilitățile navigatorului. Pentru a permite autentificarea folosind contul de *Google* sau de *Twitter* aplicația folosește modulul *Passport*, modul ce oferă un API pentru autentificarea la o mulțime de aplicații web. Pentru notificarea utilizatorilor atunci când una din sarcini expiră, aplicația folosește modulele *Nodemailer*, *Twit* și *Cron*. Modulul *Nodemailer* este folosit pentru a trimite emailuri utilizatorilor, modulul *Twit* pentru a trimite mesaje directe utilizatorilor ce s-au autentificat folosind contul de *Twitter* iar modul *Cron* este folosit pentru a notifica utilizatorii exact la data setată de ei.

Arhitectura aplicației

La baza aplicației stă transmiterea asincronă de mesaje între client și server. Deasemenea, aplicația este dezvoltată după paradigma *REST*. Aplicația este structurată pe 3 niveluri: *prezentare*, *logică*, *model*.

Nivelul *prezentare* este reprezentat de interfața aplicației. Când utilizatorii accesează aplicația aceștia pot adăuga sarcini, își pot crea un cont de utilizator sau se pot autentifica la aplicație. La crearea sarcinilor cât și la crearea unui nou cont de utilizator, aplicația validează în timp real datele introduse de ei, semnalând orice eroare. Pentru utilizatorii care nu dețin un cont de utilizator, aplicația oferă posibilitatea de a adăuga sarcini și de a fi notificați la ora și data setată(în cazul sarcinelor ce dețin un memento). Acest lucru este posibil deoarece atunci când un utilizator accesează aplicația și nu are setat niciun cookie, serverul setează un cookie unic pe termen lung pentru a putea identifica sarcinile create de utilizator, sarcini care sunt salvate în baza de date împreună cu identificatorul utilizatorului. Pentru cei care dețin cont de utilizator, pe lângă posibilitatea de a adăuga sarcini, aceștia sunt notificați prin email sau prin *Twitter*(în cazul utilizării contului de *Twitter*) și pot crea proiecte.

Nivelul *logică* este reprezentat de *controllerul* aplicației. Acesta a fost modularizat astfel încât orice eroare să poată fi găsită ușor și, deasemenea, orice schimbare să poată fi făcută ușor. Astfel, *controllerul* este format dintr-un modul ce se ocupă cu lucrul asupra bazei de date, un modul ce se ocupă cu autentificarea și un modul ce se ocupă cu comunicarea dintre client și server. Accesând aplicația, clienții inițializează două conexiuni cu serverul folosind *socket-uri*. Atunci când utilizatorul introduce date invalide la crearea unui cont de utilizator, acestuia nu-i este permis să trimită datele către server. Dacă acesta totuși ar reuși să trimită date invalide către server, va fi redirectat înapoi către pagina principală. Serverul se ocupă de validarea tuturor datelor, chiar dacă în client uneori nu sunt permise anumite acțiuni din cauza datelor invalide. Pentru comunicarea dintre client și server s-au folosit două socket-uri: un socket a fost folosit pentru transmiterea datelor de la client la server, pentru validări și notificări iar un

socket a fost folosit pentru transmiterea sarcinelor și proiectelor de la server către client la accesarea aplicației. De fiecare dată când o nouă sarcină este trimisă către server, acesta o validează și o trimite înapoi la client(în cazul în care aceasta este validă) sau trimite un mesaj de eroare clientului(în cazul în care aceasta nu este validă). Deoarece sarcinile sunt salvate în baza de date împreună cu id-ul utilizatorului care le-a creat și deoarece la introducerea unei noi sarcini datele sunt trimise folosind un socket, serverul folosește modulul *Cookie* pentru a extrage din cookie id-ul utilizatorului care a creat sarcina. În fiecare zi, la ora 0:00, serverul selectează din baza de date toate sarcinile a căror dată coincide cu data curentă și notifică utilizatorii. Utilizatorii ce s-au autentificat folosind contul de *Google* sau printr-un cont de utilizator creat din cadrul aplicației sunt notificați prin email. Cei care s-au autentificat folosind contul de la *Twitter* sunt notificați prin mesaje directe către contul de *Twitter*, deoarece cei de la *Twitter* nu pun la dispoziție adresa de email asociată contului.

Nivelul *model* este reprezentat prin baza de date **NoSQL** *CouchDB*. Deoarece serverul a fost scris în limbajul *JavaScript* utilizarea acestei baze de date a facilitat dezvoltarea aplicației. Fiecare cont de utilizator este reprezentat în baza de date printr-un document JSON. Deasemnea, sarcinile și proiectele sunt reprezentate prin documente JSON.

Detalii de implementare

Atunci când un utilizator își creează un nou cont de utilizator, serverul validează datele. Adresa de email este verificată folosind o expresie regulată. Parola trebuie să conțină cel putin 6 caractere și cel mult 30 de caractere, doar litere și numere. Deasemnea, atunci când utilizatorul confirmă parola, severul verifică dacă parolele coincid. În cazul unei erori, serverul notifică clientul iar acesta nu este lăsat să trimită formularul către server. Toate aceste validări sunt realizate în timp real, asincron.

```
socket.on('validatePass', function (data){
        var isValid = true;
        if (data.password.length < 6 || data.password.length > 32)
            isValid = false;

        data.isValid = isValid;
        socket.emit('validationResult', data);
    });
```

La autentificare, serverul verifică datele atunci când butonul *login* este apăsat. Dacă datele nu sunt valide, serverul semnalează erorile. Dacă datele sunt valide, este setat un cookie ce conține email-ul utilizatorului sau numele de cont(în cazul utilizatorilor autentificați folosind contul de *Twitter*). Pagina principală este formată dintr-un *layout* ce contine proiectele utilizatorului, un *layout* ce conține sarcinile proiectului selectat și un layout pentru notificări.

```
db.getUser(user['email'], function (resp){
        if(resp){
            getHash(user, function(user){
                if (resp.value['password'] == user['password'])
                    res.cookie("todo_logged_in",{
                        "user": resp.key,
                        "_id": resp.id
                    }, {
                        expires: new Date(Date.now()+99999999),
                        signed: true
                    });
            });

            res.redirect('/');
        } else {
          res.redirect('/');
        }
    });
```

Atunci când utilizatorul creează o sarcină sau un proiect, datele sunt transmise către client folosind un socket. Atunci când utilizatorul apasă butonul pentru a trimite datele către server, este trimis un semnal(specific fiecărei acțiuni) către server împreună cu datele ce necesită validate. În server, atunci când apare un semnal, acesta efectuează validările și emite un semnal împreună cu un mesaj de eroare(dacă datele nu sunt valide) sau cu datele, dacă datele sunt valide:

La crearea unei sarcini utilizatorul poate să seteze data, prioritatea, sarcina ce trebuie efectuată. După creare, utilizatorul poate șterge sarcina, poate să îi modifice prioritatea, să o seteze ca efectuată și să adauge notițe. Actualizarea în baza de date se efectuează în timp real: atunci când utilizatorul modifică unul dintre câmpuri, este emis către server un semnal împreună cu id-ul sarcinei și câmpul modificat(în cazul modificării) sau doar împreună cu id-ul sarcinei(în cazul ștergerii), iar serverul, la primirea acelui semnal, va realiza actualizarea sarcinei în baza de date.

Deoarece la crearea sarcinilor utilizatorii pot să introducă un memento(sau dată la care expiră sarcina), atunci când serverul este pornit, este executată o funcție care în fiecare zi, la orele 20:00(4 ore înaintea expirării), 0:00, 08:00 notifică clienții ce au de realizat sarcini la data respectivă:

```
var mailOptions = {
    from: "WebToDo++  <webtodopp@gmail.com>", // sender address
    to: resp[counter].value.loggedIn, // list of receivers
    subject: "WebToDo++ notification", // Subject line
    text: 'Hello, ' + '\n' + 'Your to do is due today: ' +
    resp[counter].value.todo
```

```
};

smtpTransport.sendMail(mailOptions, function(error, response){
    if(error){
        console.log(error);
    } else {
        console.log("Mail sent: " + response.message);
    }
});

var pm = 'Hi, your to do is due today: ' + resp[counter].value.todo;
twitterClient.post('direct_messages/new',
{'screen_name': resp[counter].value.loggedIn, 'text': pm},
function(err, reply) {
    if (!err) {
        console.log('Direct message sent...');
    } else {
        console.log(err);
    }
});
```

Utilizatorii pot sorta sarcinile în funcție de prioritate și le pot ordona după dată(în cazul sarcinilor ce au setat memento). Ordonarea se efectuează la nivel de client, folosind *jQuerry*.

Pentru împărțirea sarcinilor și dezvolatarea armonioasă a proiectului s-a folosit Github. Fiecare membru al echipei a avut un *branch* separat pe care adăuga actualizările. De obicei, fiecare membru al echipei lucra pe fișiere separate iar atunci când se lucra pe același fișier se efectua un *merge* manual al fișierului. O dată pe zi se efectua *merge* cu *master-ul*. A doua zi fiecare membru al echipei avea ultima versiune a aplicației și putea relua lucrul.

Concluzii

În concluzie, **(web to do)++** este o aplicație foarte utilă pentru organizarea timpului și a sarcinilor datorită sistemului de notificări, a simplității creării de sarcini și gestionării acestora.