Universitatea din București

Facultatea de Matematic**ă** și Informatic**ă**

LUCRARE DE LICENȚĂ

Cassandra Interface – Aplicație Web

COORDONATOR STIINTIFIC STUDENT

Conf. dr. Alin Ștefănescu Lina Luca-Cristian

Bucuresti 2019

Curpins

[Introducere 3](#_Toc10723063)

[Tehnologiile si arhitecturile folosite 5](#_Toc10723064)

[Apache Cassandra 3.11.3 5](#_Toc10723065)

[Java 8 8](#_Toc10723066)

[Spring, Spring Boot 2.0.5 si Spring Security 9](#_Toc10723067)

[Thymeleaf 14](#_Toc10723068)

[HTML si HTML5 16](#_Toc10723069)

[CSS si Bootstrap 17](#_Toc10723070)

[Javascript si Jquery 17](#_Toc10723071)

[Prezentarea aplicatiei 19](#_Toc10723072)

[Structura 19](#_Toc10723073)

[Securitate 20](#_Toc10723074)

[Client-Side 24](#_Toc10723075)

[Server-Side 33](#_Toc10723076)

[Librarii / Plugin-uri folosite 33](#_Toc10723077)

[Concluzii 34](#_Toc10723078)

[Lista figuri 34](#_Toc10723079)

[Bibliografie 34](#_Toc10723080)

Introducere

Ideea aplicatiei pleaca de la tehnologia Apache Cassandra, o baza de date NoSQL (not only SQL) destinata gestionarii cantitatilor mari de date. Din pacate interactionarea cu aceasta baza de date de catre un programator nu este foarte complexa.

Folosind limbajul CQL (Cassandra Query Language), un programator poate prelua data sau executa comenzi asupra bazei de date. Din pacate, din punct de vedere vizual si din punct de vedere al eficientei, aceasta procedura nu este una foarte benefica. Interfata consta intr-o consola care primeste comenzile CQL si returneaza rezultatul. Astfel nu exista nicio modalitate de a avea o privire de ansamblu a bazei de date fara a trece prin mai multi pasi inevitabil.

Am ales sa dezvolt aceasta aplicatie in special pentru acest motiv, de a eficientiza gestionarea acestei baze de date printr-o interfata prietenoasa, eficienta si accesibila oricui (de aici si numele aplicatiei).

Am ales ca acesta aplicatie sa fie web in primul rand pentru usurinta si accesibilitatea pe care o ofera interactiunii si comunicarii intre membri. In al doilea rand, pentru a introduce aplicatia mai usor in mediul online, care dupa parerea mea este cel mai eficient mod de a promova o creatie, mai ales aplicatii in genul acesteia.

Avand ca exemplu negativ interfata curenta a bazei de date, a fost usor de imaginat o aplicatie eficienta care sa-i ocupe locul. Astfel, comparatia dintre cele doua interfete poate fi destul de semnificativa.

\**De acum inainte voi folosi termenul de “interfata veche” si “interfata noua” pentru a face referire la interfata curenta a bazei de date Cassandra, reprezentata de consola de tip terminal, respectiv aplicatia(interfata) prezentata, Cassandra Interface.*

Spre deosebire de interfata veche, care putea fi accesata numai prin query-uri, interfata noua ofera atat facilitatea de a executa query-uri de tip text, cat si accesarea/executarea datelor prin doar cateva click-uri. Acesta contine cate o pagina speciala pentru acestea: “View/Edit”, “Console/Script”, prin care utilizatorul poate face (intr-un procent destul de mare) cam tot ceea ce poate face si interfata veche, insa mult mai usor.

Alta diferenta seminificativa, este aceea ca interfata veche nu ofera siguranta si comunicarea eficienta intre doi sau mai multi utilizatori care au acces la aceleasi date. Cassandra Interface ofera aceasta facilitate prin simplul fapt ca orice schimbare asupra bazei de date este salvata in Log si se poate accesa de catre membrii cu rol superior in orice moment. De asemenea orice schimbare de update sau delete asupra bazei de date creeaza automat un backup. Pe langa aceste lucruri, utilizatorii cu acces la aceleasi date, pot avea roluri si acces diferit asupra lor, facand totul sa se desfasoare intr-un mediu sigur. Astfel, siguranta datelor in participarea colectiva este asigurata.

Principalul scop in dezvoltarea acestei aplicatii fiind eficienizarea din punct de vedere vizual si temporal, Cassandra Interface ofera acest lucru prin fiecare componenta a aplicatiei. Fiecare pagina in parte ofera ceva aparte utilizatorului, care poate beneficia ce acel lucru intr-un timp cat mai scurt.

Pentru a putea beneficia de calitate in dezvoltarea acestei aplicatii atat din punct de vedere al programatorului cat si din perspectiva unui utilizator, am folosit tehnologii complexe in acest sens.

Din motive evidente, baza de date folosita pentru aplicatie este Apache Cassandra, impreuna cu limbajul CQL.

Ca limbaj principal de programare pentru backend am folosit Java 8, fiind unul dintre cele mai bune tehnologii in dezvoltarea aplicatiilor web. De asemenea, este si o preferinta subiectiva. Impreuna cu acesta, am folosit framework-ul Spring Boot integrat pentru Java, oferind o usurinta si o calitate mai mare in dezvoltarea aplicatiei. Pentru o securitate eficace am folosit o tehnologie derviata din Spring si anume Spring Security.

Comunicarea intre backend si frontend este realizata prin tehnologia Thymeleaf, o librarie din Java, care aplica transformari si proceseaza template-urile afisand datele produse de aplicatie. De asemenea, a fost posibila integrarea cu Spring Security si crearea template-urilor de tip layout prin adaugarea unor dialecte la aceasta librarie.

Pentru frontend, fiind o aplicatie web, am folosit HTML5, impreuna cu elementele de design alte CSS-ului si Bootstrap-ului si crearea de scripturi in Javascript si Jquery.

In continuare urmeaza descrierea in amanunt a tehnologiilor folosite si a aplicatiei.

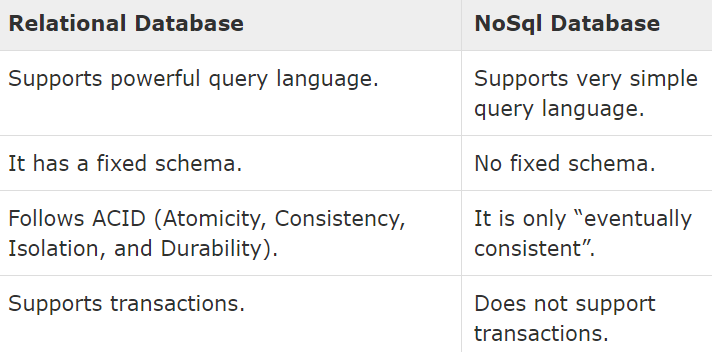
Tehnologiile si arhitecturile folosite

Apache Cassandra 3.11.3

Apache Cassandra este o baza de date “open-source”, distribuita (baza de date in care nu toate dispozitivele de stocare sunt atasate la acelasi procesor, fiind posibila stocarea pe mai multe calculatoare fie locale, fie dispersate intr-o retea de calculatoare interconectate [15]), non-relationara (NoSQL) de tip “wide column store”, putand fi interpretata ca o depozitare cheie-valoare in doua dimensiuni [16], creata pentru putea face fata unui numar foarte mare de date [3].

Avinash Lakshman si Prashant Malik au dezolvtat initial Cassandra la Facebook pentru a imbunatati motorul de cautare, apoi Facebook a lansat oficial Cassandra ca proiect “open source” la codul Google in iulie 2008 [3].

O baza de date NoSQL prezinta un mod de a depozita si recupera date diferit fata de mecanismul traditional al bazelor de date relationare, avand avantajul procesarii unui numar foarte mare de date mult mai eficient [17].



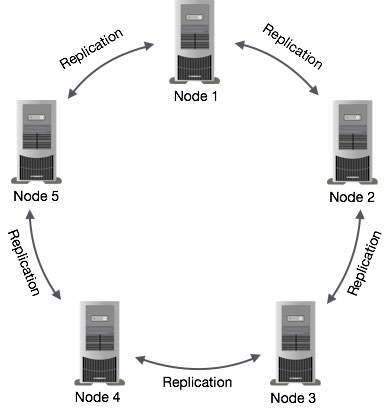
[17]

Cassandra a devenit populara datorita numeroaselor caracteristici de care dispune, cum ar fi capacitatea foarte mare de scalabilitate, avand posibilitatea de adaugare de echipament pentru mai multe date. Pe langa aceasta, Apache Cassandra este facuta in asa fel incat nu poate avea niciun punct de eroare, fiind disponibila in orice moment. De asemenea, cu cat baza de date este mai complexa in arhitectura cu atat viteza este mai buna.

Spre deosebire de alte baze de date NoSQL, Cassandra a fost dezvoltata pentru “fast writes”, fiind capabila sa stocheze sute de tera de date fara a ingreuna citirea acestora. Un alt atribut este si flexibilitatea tipruilor de date, putand depozita toate formatele de tipuri de date, incluzand cele structurate si semi-structurate.

Arhitectura consta intr-un sistem “peer-to-peer” distribuit de-a lungul mai multor noduri (in care sunt distribuite la randul lor datele), aceste noduri apartinand de un cluster (grup). Un cluster contine unul sau mai multe centre de date (un centru de date este format din mai multe noduri). Fiecare nod dintr-un cluster are acelasi rol si este independent si in acelasi timp interconectat cu celelalte noduri. Atributul principal este ca fiecare nod poate accepta cereri de scriere/citire, iar daca un nod pateste ceva, aceste cereri pot fi satisfacute de celealte noduri.

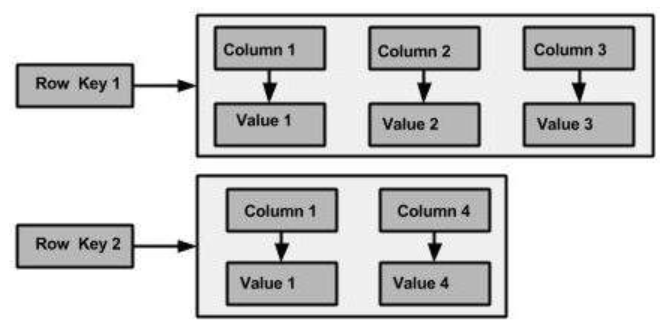
Cassandra lucreaza prin replicarea datelor, astfel unul sau mai multe noduri dintr-un cluster contine replicari pentru o anumita bucata din datele din acel cluster. Fiecare cerere de scriere este captata de un log din fiecare nod, apoi detele sunt trimite la “mem-table”, un tabel de memorie unde sunt trimise temporar datele, inainte de a fi partitionate si replicate in cluster. Daca “mem-table” este plin, datele se vor stoca de asemenea temporar in “SStable”, un fisier pe disk, pe care Cassandra il va interoga periodic pentru a elimina datele inutile. Imaginea de mai jos arata o schema de noduri intr-un cluster si cum functioneaza replicarea astfel incat sa nu existe niciun fel de eroare [17].



[17]

Termenul de keyspace reprezinta un container pentru o lista de una mai mai multe “column families” (structura in care sunt depozitate datele – tabel). Ca si atribute, un keyspace poate fi determinat atat de un factor de replicare, care reprezinta numarul de copieri a datelor in clusterul din care face parte, cat si de strategia pentru a replica aceste date.

O familie de coloane la randul ei este un container de colectii de linii, fiecare linie continand coloane ordonate, fiecare linie avand cel putin o coloana ca facand parte din “partion key” (cheia primara). Cheia primara poate fi simpla sau compusa (din mai multe coloane) si este responsabila pentru distribuirea datelor in partitia curenta, astfel cautarea se va face implicit dupa aceasta cheie. Pentru o cautare in care nu este mentionata cheia primara, performanta va scadea senimificativ; in acest caz, pentru unele coloane care nu fac parte din cheia primara, se introduce termenul de index pe coloana respectiva, adaugandu-i anumite atribuite pentru a facilita cautarea. Pe langa cheia primara mai exista si coloane “clustering key”, care sunt reponsabile pentru sortarea datelor in partitie. Figura de mai jos reprezinta schema unei familii de coloane [17].



[17]

Apache Cassandra ofera un limbaj ca suport sub forma unui terminal pentru comunicarea intre user si baza de date, si anume CQL (Cassandra Query Language), avand posibilitatea de a crea keyspace-uri, insera date sau executa diferite query-uri [17]. Atat sintaxa limbajului cat si tipurile de date seamana destul de mult cu cea de la bazele de date relationare cum ar fi SQL, astfel exista comenzi asemanatoare pentru: selectare, creare, alterare, stergere, inserare, actualizare, etc. Pe langa tipurile de date cunoscute cum ar fi: text, int, boolean, date, etc, Cassandra introduce termeunul de “frozen”, un tip de data care accepta atat tipuri create de utilizator (UDT – user data type), cat si colectii cunoscute ca liste, seturi si harti. Ca si in alte baze de date, aceasta ofera posibilitatea crearii de functii si triggere asupra tabelelor. Mai mult decat atat, cautarea fiind restrictionata de cheia primara, se poate crea un “materialized view”, adica un alt tabel conectat cu tabelul de baza, in care pe langa cheia primara inititala, se adauga si alte coloane la aceasta, facand posibila cautarea dupa acestea (fara a altera tabelul initial, iar modificarile sunt sincronizate). Mai jos sunt prezentate cateva exemple de sintaxa pentru acest limbaj.

CREATE KEYSPACE IF NOT EXISTS admin WITH REPLICATION = { ‘class’ : ‘SimpleStrategy’, ‘replication\_factor’ : 3 } AND DURABLE\_WRITES = true;

CREATE TABLE IF NOT EXISTS admin.users (user\_id text PRIMARY KEY, age int, name frozen<fullname>, data list<text>);

CREATE TYPE admin.fullname (first\_name text, last\_name text);

INSERT INTO admin.users (id,age) values (‘1’,21);

CREATE MATERIALIZED VIEW IF NOT EXISTS admin.users\_by\_age

AS SELECT \* FROM admin.users

WHERE user\_id IS NOT NULL AND age IS NOT NULL

PRIMARY KEY (age, user\_id);

SELECT (age,user\_id) FROM admin.users\_by\_age;

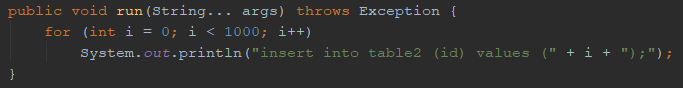
Motivul pentru care am ales aceasta baza de date este prezentat in Introducere.

Java 8

Java este un limbaj de programare orientat pe obiecte, creat de catre James Gosling la inceputul anilor ’90, fiind lansat in 1995. Este un limbaj cu o intindere larga, fiind folosit in numeroase aplicatii distribuite, incluzand cele mobile.

In sintaxa limbajului se regasesc o mare parte din elemente de C sau C++, dar modelul obiectelor este mai simplu si are mai putine facilitati de nivel jos. Ca de exemplu, Java nu are supraincarcare operatorului, iar mostenirea multipla este valabila doar pentru interfete nu si pentru clase. Avantajul major este ca un program Java corect si compilat, poate fi rulat fara modificari pe orice platforma care are instalata o masina virtuala Java. Acest lucru este posibil deoarece sursele sunt compilate intr-un format numit cod de octeti, acesta fiind independent de tipul platformei.

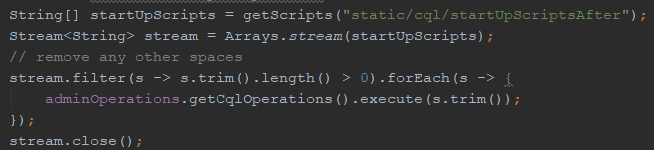
Pentru a demonstra aceste referinte, urmeaza o imagine sustrasa din proiect, care contine un bloc de cod care se apeleaza la pornirea serverului afiseaza in consola acestuia (System.out.println) 1000 de linii (for de la 0 la 999), fiecare continand un query de inserare pentru baza de date. Acest bloc a fost folosit pentru testare si nu are un rol functional in aplicatie.



Dintre cele 4 platforme furnizate de Oracle (Java Card, Java ME – micro edition, SE – standard edition si EE – Enterprise Edition, aplicatia Cassandra Interface foloseste versiunea 8 a platformei standard, care este destinanta sistemelor de tip workstation [4].

Java SE 8 a aparut pe 18 martie 2014 si este considerata cea mai mare actualizare, modificarile asigurand faptul ca Java ramane o prioritate la nivelul aplicatilor de intreprindere. Spre deosebire de celelalte versiuni, acesta aduce imbunatatiri seminificative.

Printre acestea se numara expresiile lamda, care sunt considerate cele mai mari caracteristici in Java 8. Aceste expresii actioneaza ca o functie anonima, intr-o bucata de cod foarte scurta, ajutand astfel si la coding style. De asemenea, tot in acesta versiune au fost introduse stream-urile, o alta actualizare majora, care vine cu diferite tipuri de operatiuni de flux (filtrare, sortare, etc), folosite in mod eficient impreuna cu expresiile lamda [5]. Urmeaza o imagine din aplicatie care contine un astfel de exemplu, care pentru un vector de stringuri, il filtreaza, eliminand stringurile vide, iar pentru fiecare dintre acestea, executa o anumita comanda.



Am ales acest limbaj in primul rand pentru performanta pe care o ofera atat ca viteza si usurinta de scriere, dar si pentru calitatea dovedita de-a lungul anilor, fiind platforma pentru milioane de dispozitive. In al doilea rand pentru diversitatea de care dispune acest limbaj, avand posibilitatea de integrare cu aproape orice domeniu.

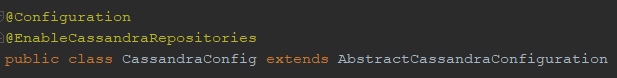
Spring, Spring Boot 2.0.5 si Spring Security

Spring este un framework de aplicatii pentru platforma Java, iar prima versiune a sa aparut pe 1 octombrie 2002 si a fost scrisa de Rob Johnson. Framework-ul poate fi folosit de orice aplicatie Java, insa principalul sau atribut faciliteaza dezvoltarea aplicatiilor web.

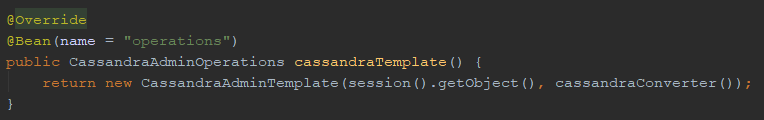
Specific pentru Spring este container-ul IoC (inversion of control), care produce configuratii si se ocupa de obiectele din Java cu ajutorul reflexiei (abilitatea unui proces de a-si examina si modifica propria structura si comportament [8]). Container-ul se ocupa de organizarea duratei de viata (creare, apelare, configurare, etc) a obiectelor care se mai numesc “beans” si poate fi configurat cu ajutorul unui fisier XML (Extensible Markup Language) sau prin detectarea unor adnotari specifice asupra claselor de configurare. Aceste obiecte la randul lor pot fi obtinute printr-o cerere catre container pentru un obiect cu un anumit nume sau de un anumit tip (dependency lookup) sau prin-o metoda care trimite obiectul dupa nume intr-un alt obiect (dependency injection) [6]. Imaginea de mai jos prezinta a doua metoda, in care obiectul (bean-ul) UserService creat si stocat in contextul Spring, este “injectat” intr-un obiect intr-o clasa cu ajutorul adnotarii @Autowired.



Spring are propriul AOP (aspect-oriented programming), care modularizeaza relatiile de tip cross-cutting in aspecte. Cu alte cuvinte, spre deosebire de modulele traditionale care incapsuleaza obiectele si contin codul necesar pentru ca acestea sa implementeze serciviile sitemului, modulele AOP contin codul necesar implementarii unui singur serviciu a sistemului, considerant toate obiectele posibile [9]. In imaginea urmatoare, se poate observa o clasa pe care framework-ul o va recunoaste ca fiind una de configurare si o va manipula in consecinta (o va instantia ca un bean) datorita adnotatiei corespunzatoare.



Alta caracteristica importanta este accesarea datelor in Spring. Acesta ofera suport pentru cele mai populare framework-uri de acces de date din Java cum ar fi Cassandra (in cazul de fata), JDBC, Hibernate, JDO, JPA, etc. De asemenea, tranzactiile au o diversitate destul de mare, de la tranzactii locale si globale pana la tranzactii in aproape toate mediile de platforme din Java [6]. Imaginea de mai jos arata crearea unui Bean pentru un obiect dintr-o clasa de configurare a bazei de date Cassandra si anume CassandraAdminTemplate, prin care se va permite executarea comenzilor si preluarea datelor asupra bazei de date.



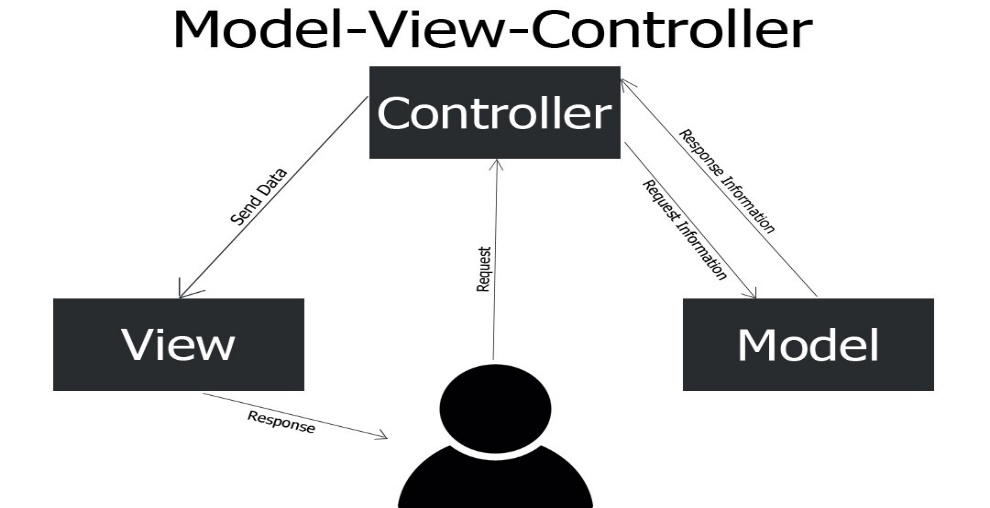
Framework-ul Spring foloseste ca arhitectura propriul MVC (Model-View-Controller) pentru aplicatii web. Aceasta arhitectura este folosita in mare parte pentru dezvoltarea interfetelor pentru utilizatori si imparte aplicatia in trei parti interconectate:

Model, componenta centrala, independenta de interfata si care se ocupa de logica si regulile cu privire la datele aplicatiei (baza de date);

View, reprezentand informatia vizuala pe care o vede utilizatorul, partea de frontend;

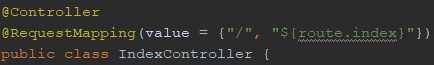
Controller, care primeste intrari si trimite mai departe catre model sau view.

Cu alte cuvinte, modelul este responsabil pentru organizarea datelor aplicatiei si primeste intrari din view cu ajutorul controllerului, view-ul reprezentand astfel prezentarea modelului intr-un format aparte, iar controllerul raspunde intrarilor din view, optional le valideaza si actioneaza in consecinta asupra modelului [10].

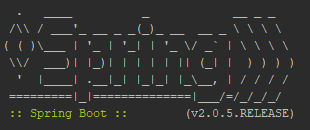


[11]

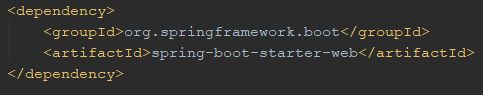
Figura urmatoare arata cum framework-ul va recunoaste, printr-o adnotare, o clasa ca fiind o componenta a arhitecturii MVC si anume controller si va trata intreaga clasa cu componentele ei ca atare.



Spring Boot reprezinta o solutie bazata pe o conventie de configuratii considerate ca fiind unele dintre cele mai bune din platforma Spring pentru crearea de aplicatii “stand alone”, aplicatii care pornesc de la o configurare minima si eficienta pentru a putea rula cat mai rapid (“just run”). Acesta ofera simplitate, configurand automat cand este cazul si folosind Project Object Models (POMs), simplificand astfel configuratiile Maven [6].



Un POM este un fisier XML care contine informatii despre proiect si despre configuratiile ce contribuie la constructia acestuia. In acest fisier pot fi incluse toate dependintele proiectului, plugin-urile, versiunile acestora si multe altele, oferind astfel atat simplitate cat si complexitate in dezvoltarea aplicatiei [13]. Imaginea urmatoare reprezinta o dependinta Maven din POM-ul proiectului care permite integrarea web cu framework-ul Spring Boot.

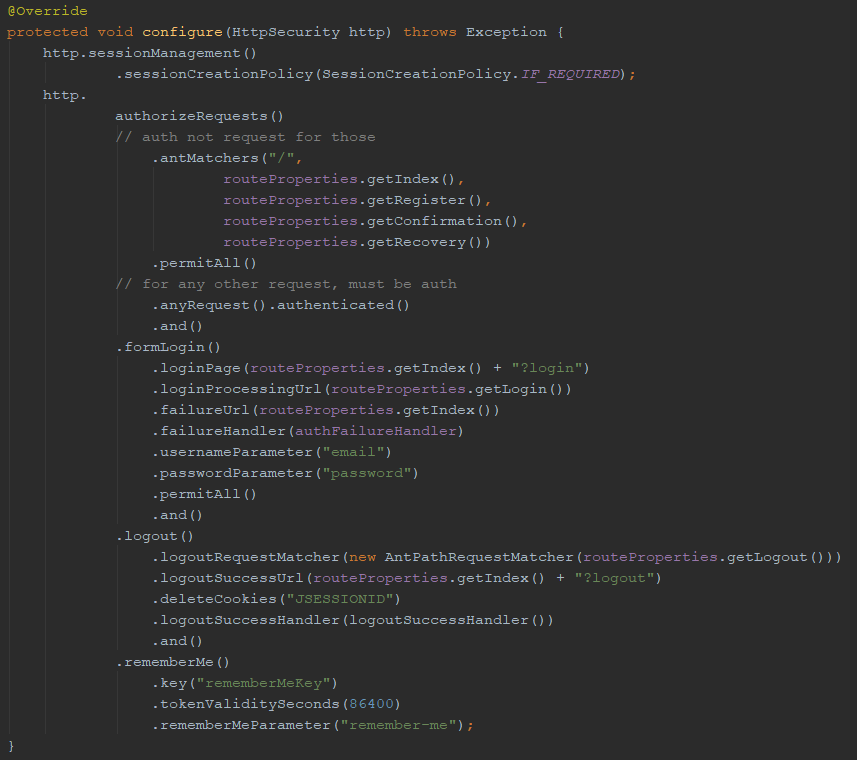


Spring Boot cuprinde implicit si server-ul pe care va rula aplicatia, si anume Tomcat. Apache Tomcat este un mediu de server web in care un program in Java poate rula, implementand implicit cateva specificatii din Java EE incluzand Java Servlet, JSP (JavaServer Page), Java EL si WebSocket [12], fiecare facand parte din componentele unei aplicatii web. Figura urmatoare arata initializarea server-ului la pornirea Spring Boot-ului.

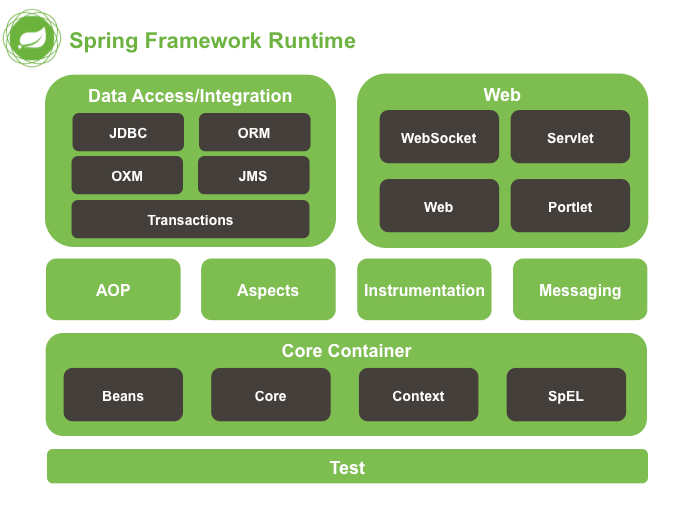


Spring Security este un framework care aduce conceptul de autentificare si autorizare pentru securizarea aplicatiilor Java. Autentficiarea reprezinta procesul de identificare a utilizatorului ce vrea sa acceseze aplicatia, iar autorizarea se ocupa de accesul in executarea actiunilor asupra ei (cereri web, metode, acces pe un anumit domeniu, etc). Alte avantaje constau in suportul oferit pentru configurarile in Java, posibilitatea de integrare cu Spring MVC, portabilitate, CSRF (Cross-site request forgery) protection (diferite atacuri asupra aplicatiei prin comenzi neautorizate provenite de la un utilizator in care aplicatia are incredere [20]), etc [19].

Imaginea de mai jos reprezinta functia de configurare a framework-ului pentru Cassandra Interface. Acesta configurare consta in restrictionarea accesului asupra paginilor web ale aplicatiei utilizatorilor care nu sunt autentificati; in acest caz, accesul lor va fi limitat la pagina de pornire si la cele care tin de autentificare. De asemenea apar si setarile subiective pentru pagina de login si logout (urls, handlers, parametrii); in plus, se configureaza si facilitatea de “remember-me”, care memoreaza user-ul la autentificare pentru un anumit interval de timp.



Am ales acest framework pentru performanta si complexitatea de care dispune, fiind unul dintre cele mai folosite framework-uri pentru aplicatii web in Java.



[14]

Thymeleaf

Thymeleaf este o librarie din Java pentru template-uri XML/XHTML/HTML5, care poate rula atat pe un mediu web cat si pe un mediu non-web, aplicand transformari asupra acestor template-uri in urma carora se pot afisa date produse de aplicatie [21], permitand astfel “comunicarea” intre backend si frontend.

Arhitectura Thymeleaf permite procesarea rapida prin parsarea fisierelor astfel incat sa se execute cat mai putine operatii.

O puternica unealta a acestei librarii o constituie extensibilitatea acestuia, permitand definirea anumitor noduri (DOM – Document Object Model), care impreuna cu alte “artefacte”, formeaza un dialect [2]. Thymeleaf ofera default dialectul Standard, insa mai pot fi adaugate si altele.

In imaginea de mai jos se intampla urmatoarele: prin aplicarea precompilarii Thymeleaf cu dialectul Standard, acest element de HTML (<div>) va fi atasat paginii, numai in cazul in care in request va exista obiectul param.login (th:if). In caz afirmativ, continutul div-ului va fi inlocuit cu mesajul corespunzator cheii “login.redirect.message” din fisierul de mesaje al aplicatiei (th:text).



Alt dialect foarte folosit este cel pentru Layout, care permite “decorarea” paginilor dupa un anumit layout, fiind foarte eficient pentru a nu duplica aceasi cod pentru fiecare pagina in parte.

Prin aceasta imagine este reprezentata sintaxa dialectului Layout care spune librariei ca acesta pagina HTML este layout pentru alte pagini.



Iar prin acesta sintaxa, Thymeleaf va stii ca acest bloc HTML este suport (fragment) pentru pagina care va “decora” acest layout. Astfel, daca intr-o pagina care foloseste acest layout exista acelasi fragment dar cu continut, acel continut va fi inlocuit in pagina de layout in fragmentul respectiv.



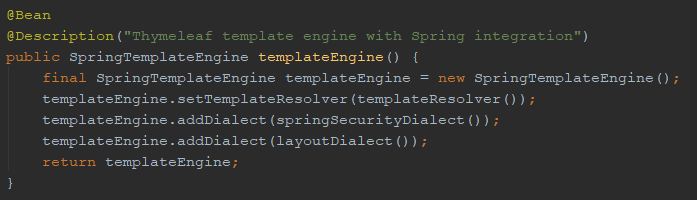
Dialectul pentru Spring Security este de asemenea folosit. Prin integrarea cu acest framework, Thymeleaf va putea avea acces la statusul legat de autentificare si autorizare a utilizatorului care acceseaza pagina. Astfel se pot impune restrictii direct asupra blocurilor de html din frontend, in functie de accesul pe care il are la momentul respectiv.

Imaginile de mai jos fac referire la sintaxa dialectului care restrictioneaza blocul respectiv de html (<li>) utilizatorilor care sunt autentificati, respectiv utilizatorului (autentificat si el) cu rol de ADMIN.





Ca ultim exemplu, in figura de mai jos este un fragment din configurarea din Cassandra Interface a librariei Thymeleaf, in care se creeaza bean-ul (integrare cu Spring) pentru motorul care va fi folosit in compilarea template-urilor, unde se adauga cele doua dialecte (cel standard este inclus automat).

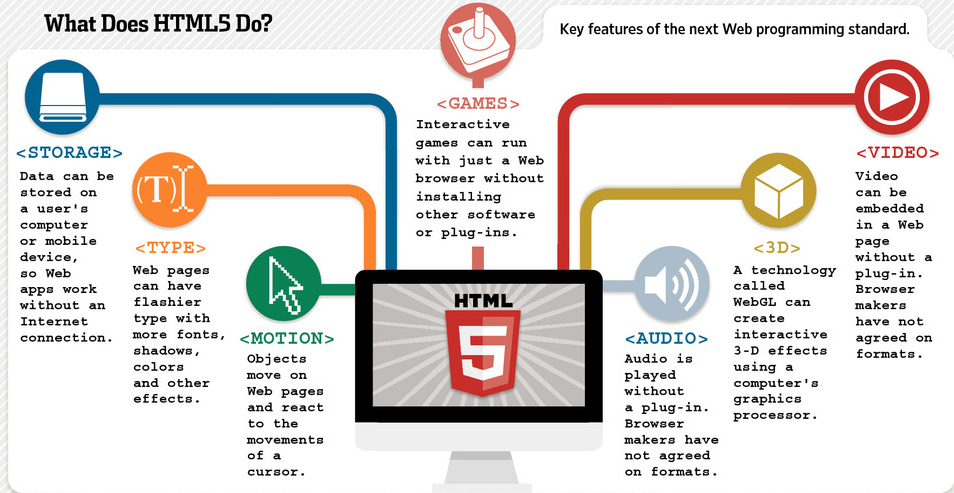


HTML si HTML5

HTML (Hype Text Markup Language) este un limbaj de marcare (metoda de formatarea unui text intr-o pagina web) folosit pentru crearea site-urilor web. Browserele web primesc documente HTML fie din spatiul local fie de la un server web si le parseaza in pagini web.

HTML creeaza documente structurate folosind blocuri pentru texte cum ar fi paragrafe, liste, link-uri, imagini si alte obiecte de tip bloc. Acestea sunt delimitate prin tag-uri ca <p> </p> pentru paragrafe sau <img> </img> pentru imagini. O pagina web este delimitata de tag-ul <html> </html>, iar continutul vizibil pentru user este delimitat de tag-ul <body> </body> [22]. Astfel, intreg documentul este o structura arborescenta, fiecare tag putand include alte tag-uri. De asemenea fiecare tag in parte poate avea informatii suplimentare (atribute) despre obiectul respectiv, informatii pe care HTML le va folosi in parsarea documentului.

HTML5 este a cinea si cea mai buna versiune a limbajului HTML, aparut pe 22 ianuarie 2008 si are ca scop imbunatatirea limbajului astfel incat sa poata avea suport pentru cele mai noi fisiere multimedia si alte caracteristici, fiind in acelasi timp citibil atat pentru user dar si pentru calculatoare. Faciliteaza si incorporarea cu diferite API-uri pentru aplicatii web mai complexe [23]. Pentru a declara documentului ca este folosit HTML se foloseste intaintea tag-ului <html>, tag-ul <!DOCTYPE html>.



[24]

CSS si Bootstrap

CSS (Cascading Style Sheet) este un limbaj de stilizare pentru prezentarea unui document scris intr-un limbaj de marcare (HTML). Numele de “cascading” vine de la puterea limbajului de a determina prioritatea regulilor (pe care sa o aleaga) in caz ca exista mai multe.

CSS-ul are o sintaxa simpla, astfel ca un fisier de stilizare (style sheet), contine o lista de reguli, fiecare regula avand unul sau mai multi selectori si un bloc de declarare. Selectorul determina carei parti din document ii este aplicata stilizarea.

Mai jos este un exemplu de sintaxa a CSS-ului, care seteaza culoarea rosie si textul bold pentru continutul tuturor paragrafelor care au clasa “paragraf” dintr-un document, “p.paragraf” fiind selectorul, iar ceea ce se afla intre acolade este blocul de reguli.

p.paragraf {

color: red;

font-weight: bold;

}

Boostrap este un framework “open source” pentru CSS. A aparut initial sub numele de Twitter Blueprint, fiind creat de catre Mark Otto si Jacon Thornton pentru platforma Twitter. Au aparut apoi mai multe versiuni, iar Boostrap 3 (cel folosit in Cassandra Interface) s-a stabilit pe 19 august 2013, urmat apoi de Boostrap 4 pe 29 octombrie 2014 in versiunea alpha si finalizat pe 18 ianuarie 2018 [26].

Boostrap este format in mare parte din o colectie de reguli CSS si optional sripturi de JavaScript, care folosite si apelate sub forma unor atribute ale elementelor dintr-un document, transforma elementele respective si le stilizeaza pe baza regulilor definite de framework.

Astfel, avand posibilitatea de a modifica numai acele atribute dupa bunul plac al utilizatorului, stilizand prin acestea documentul, Boostrap devine un framework in acelasi timp complex, dar si maleabil si usor de folosit, economisind astfel timp si cod.

Javascript si Jquery

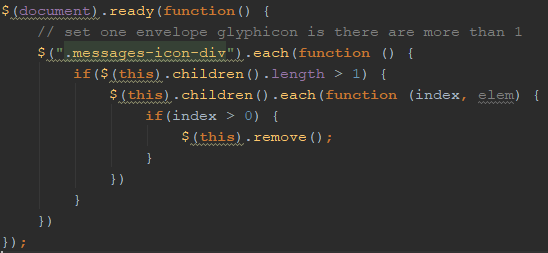
Javascript este un limbaj de programare interpretat (intructiunile sunt executate direct, fara compilare), orientat pe obiecte bazate pe prototipe (obiectele se pot refolosi). Alaturi de CSS si HTML, Javascript este unul dintre tehnologiile nucleu ale World Wide Web-ului si este o parte esentiala pentru aplicatiile web.

Chiar daca sunt unele asemanari intre Javascript si Java din punct de vedere al sintaxei si al librariilor, acesta a fost influentat in mare parte de limbaje de programare ca Self si Scheme. De asemenea serializarea JSON este bazata pe JavaScript [27].

O mare caracteristica a limbajului o consituie facilitatea de a putea comunica atat in partea de client (client-side), cat si in partea serverului (server-side), creand astfel o comunicare eficienta intre frontend si backend. O pagina web poate avea sau nu, unul sau mai multe fisiere (scripturi) javascript, fiecare rulandu-se atunci cand pagina web este incarcata. Unele dintre aceste scripturi creeaza triggere, parti de script care se vor apela atunci cand un anumit eveniment este intalnit in pagina.

Jquery este o librarie din JavaScript creata pentru a simplifica manipularea elementelor DOM din cadrul paginii HTML, a evenimentelor, animatiilor, cererilor ajax, etc, facand navigarea intr-un document mult mai usoara [28]. Astfel, scriptul scris in Jquery este mult mai scurt si mai lizibil decat unul scris in Javascript, incurajand in acest sens coduri mai complexe.

Imaginea de mai jos demonstreaza complexitatea sintaxei Jquery. Odata cu incarcarea paginii, scriptul se ruleaza (apelandu-se functia de incarcare completa a paginii ($(document).ready()), apoi prin Jquery, preiau toate elementele gasite cu care contin clasa “messages-icon-div” si pentru fiecare element gasit, daca are mai mult de un element-copil, sterg acele elemente pana ramane unul singur. Astfel scriptul elimina duplicatele din interioriul unui element prin doar cateva comenzi simple.



Prezentarea aplicatiei

Structura

Aplicatia pe partea de frontend, este structurata in aproximativ 4 parti importante:

Prima este pagina de Home, la care au acces toti utilizatorii, fie ei autentificati sau nu. In cadrul acestei pagini este prezentata o descriere a aplicatiei, mai mult decat atat, de pe acesta pagina utilizatorii se pot autentifica sau inregistra in aplicatie.

A doua parte consta in pagina de Profil a membrului autentificat cu succes, care este disponibila spre a vizualiza si a aduce modificari in cadrul contului acestuia.

A treia parte si cea mai reprezentativa o reprezinta paginile cu interactiune in baza de date. De asemenea aceste pagini pot fi accesate numai dupa ce membrul este autenificat si pe langa acest lucru, baza de date in sine are autentificare si roluri de acces proprii. Aceste pagini permit utilizatorului cu acces sa vizualizeze / creeze / editeze / exporte / caute in baza de date, in functie de nevoile acestuia si de setarile facute.

A patra parte consta intr-o consola universala cu acces la baza de date, care este disponibila numai membrilor cu acces de ADMIN global, neavand legatura cu baza de date personalizata pentru fiecare utilizator.

Toate aceste pagini sunt expuse membrilor sub forma unui meniu mereu prezent si usor de utilizat. De asemenea, folosind Thymeleaf (descris mai sus), aplicatia foloseste un layout cu mai multe fragmente, astfel paginile propriu-zise sunt construite dinamic, fara replicare de cod.

Pe partea de backend, aplicatia este structurata in 2 parti esentiale:

Prima este partea de configurare, care contine toate clasele care configureaza dependintele proiectului, cat si clasele de proprietati, validari, si tot ce tine de partea “fizica” a aplicatiei.

A doua parte este cea de web, in care se afla controllerele, modelele, serviciile, event-urile, si tot ceea ce tine de o aplicatie REST, partea de comunicare cu frontend-ul si cu baza de date.

Toate acestea vor fi explicate la capitolul de Server-Side.

Securitate

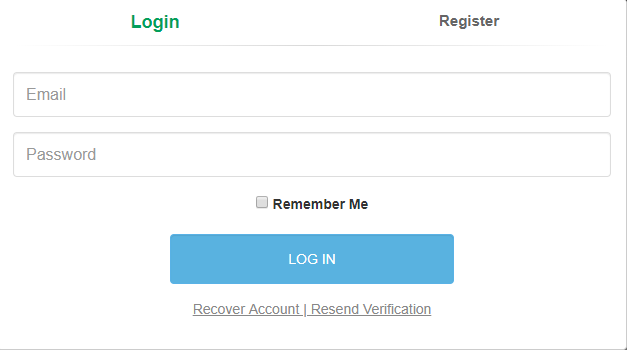
Cassandra Interface a fost dezvoltata cu scopul de a simplica anumite procese si de a prezenta o noua metoda de organizare a datelor intr-o baza de date NoSQL. Astfel, inevitabil, platforma este dependenta de baza de date pentru care ruleaza, devenind nefunctionala in cazul in care Apache Cassandra nu este activa.

Fiind o aplicatie web, aplicatia se bazeaza pe conceptul de REST, cereri facute de catre utilizatori catre server (controller), la care se trimite un raspuns adecvat actiunii lor. De asemenea, din cauza faptului ca mai multi utilziatori pot accesa in acelasi timp aplicatia, respectiv baza de date, atat eficienta cat si securitatea trebuie sa fie pe masura. Astfel, o conexiune a utilizatorului cu aplicatia este legata de sesiunea acestuia in browser-ul web, devenind inactiva odata cu lipsa activitatii din cadrul aplicatiei in sesiunea curenta.

In mare parte, Spring Security se ocupa de securitatea la nivelul aplicatiei, insa cea la nivelul accesarii bazei de date a trebuit construita manual. Mai departe este descris modul in care functioneaza securitatea in Cassandra Interface.

Figura de la pagina 13, care descrie configuratia framework-ului Spring Security in cadrul aplicatiei, expune cat de poate de simplu puterea acestuia. Utilizatorii sunt restrictionati din a accesa anumite pagini in functie de starea lor de autentificare si autorizatie. Pe langa aceste setari, Spring Security se ocupa si “retinerea” unui utilizator autentificat in contexul aplicatiei, pentru a cunoaste in orice moment datele acestuia. Astfel, urmeaza o prezentare a modului de autentificare a unui user si cum este retinut acesta in context.

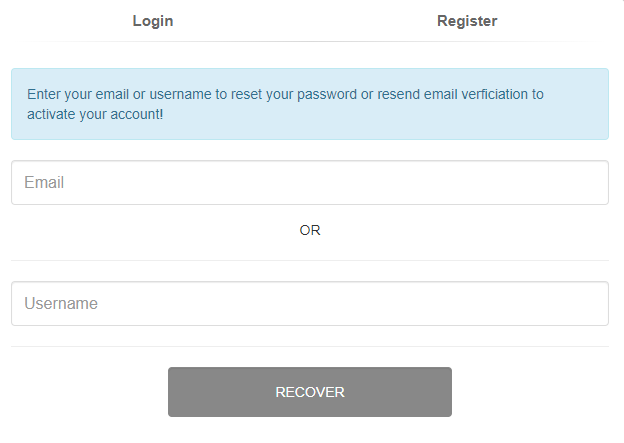
In imaginea urmatoare este prezentat formularul de login in aplicatie. Acesta apare sub forma unui “modal” in pagina, la apasarea butonului de Login din meniul alicatiei.



Dupa cum se poate observa, autentificarea unui utilizator deja inregistrat se face dupa email-ul si parola acestuia. De asemenea exista posibilitatea de a bifa “remember-me”, facilitate garantata de catre Spring Security, mentinand utilizatorul logat chiar daca acesta paraseste aplicatia. In caz de eroare la autentificare, aplicatia afiseaza mesaje corespunzatoare, ca de exemplu:



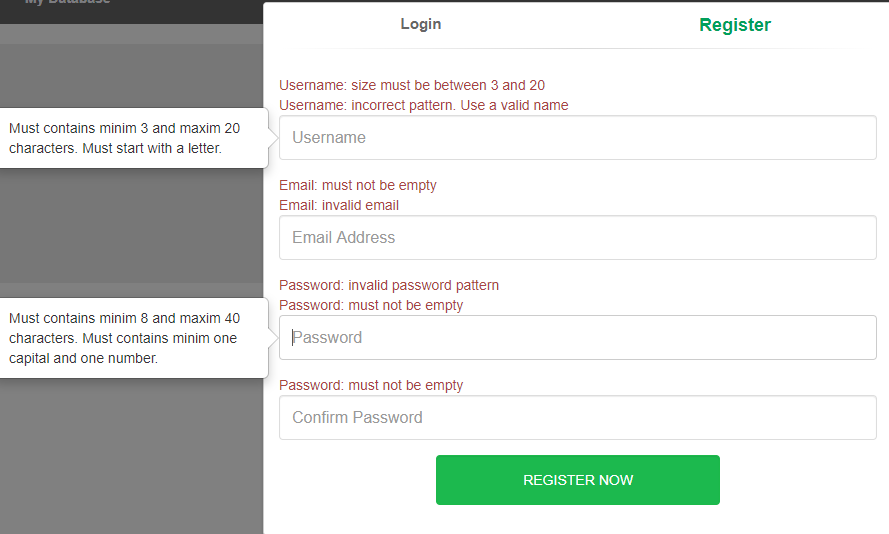
In cazul pierderii contului sau a neactivarii acestuia, user-ul are posibilitatea de a recupera parola prin email sau a retrimiterii email-ului de activare (vom discuta despre asta la inregistrare) printr-un click in subsolul formularului de logare (Recover Account…), care va deschide alt modal cu urmatorul formular:



Intern, aplicatia isi va da seama daca utilizatorul doreste sa-si reseteze parola sau sa rerimita email-ul de activare prin testarea starii contului respectiv.

Revenind la autentificare, daca acesta a avut succes, datele din formular sunt transmise catre Spring Security, care ofera posibilitatea de customizare al acestui proces, lasand programatorul sa decida care anume din datele utilizatorului sa se pastreze in context si sub ce forma. Astfel, aplicatia Cassandra Interface, preia datele (in CassandraUserDetailsService), verifica daca user-ul exista si il preia din baza de date, urmand apoi sa se stabileasca rolurile acestuia in cadrul aplicatiei si sa se returneze aceste date mai departe catre framework (CassandraUserDetails, care implementeaza clasa default UserDetails). In orice alt caz, se retruneaza o eroare sub forma unui mesaj reprezentativ pentru utilizator.

Pe langa acestea, securitatea nu trebuie sa se opreasca aici, urmeaza o imagine cu formularul de inregistrare in aplicatie (disponibil in pagina de Home, in meniul aplicatiei), iar mai departe descrierea sistemului.

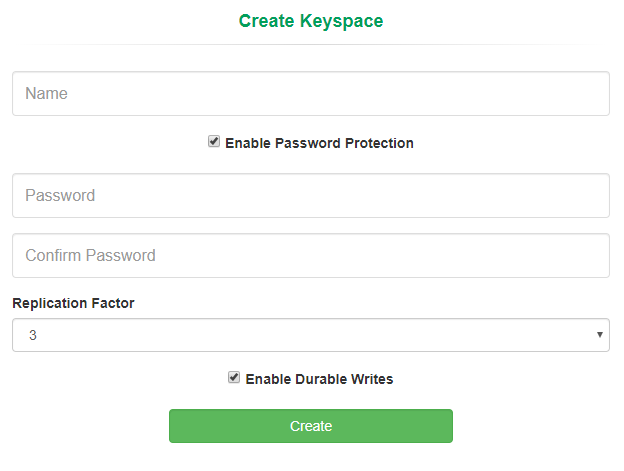


Pentru a se putea observa sistemul de erori, imaginea de mai sus este facuta dupa trimiterea formularului gol catre server, acesta incercand sa-l valideze, insa fara succes, returnand astfel erorile specifice. De asemenea, aplicatia ofera cateva popover-uri cu informatii legate de validarea datelor care trebuie introduse. Inregistrarea consta intr-un email si username unic global si o parola cu un minim de dificultate.

Informatiile sunt transmise la controllerul de register, unde sunt validate. Daca nu exista erori, se creeaza un user cu aceste date, se genereaza un ID, se cripteaza parola, se seteaza rolul de USER si se adauga in baza de date ca utilizator inactiv. Totodata se genereaza un link care se trimite pe email-ul cererii de inregistrare, care contine un token de active a contului. Daca utilizatorul va incerca sa se logheze fara accesarea acelui link din email, va primi un mesaj corespunzator de eroare. Astfel, confirmarea prin email a contului reprezinta o metoda eficienta de securitate impotriva crearii de conturi invalide si validarea integritatii utilizatorului.

Mai departe vom vorbi despre securitatea in cadrul interactiunii cu baza de date.

Un utilizator inregistrat si logat cu succes, are acces la pagina personala in baza de date indiferent de rolul pe care il detine (prin apasarea butonului “My Database” din meniul aplicatiei). Platforma ii ofera utilizatorului posibilitatea sa-si creeze propriul keyspace (structura descrisa in capitolul anterior), protejata cu parola sau nu. Figura de mai jos prezinta formularul pentru acest proces (tot sub forma unui modal-view).



Se poate observa ca un keyspace, poate fi sau nu securizat prin parola. Creatorul astfel are alegerea libera daca doreste sa-si protejeze sau nu keyspace-ul. Odata creat, acesta este stocat atat in baza de date a userilor ca facand parte din colectia creatorului, cat si in baza de date a keyspace-urilor, cu setarile alese in formular. Astfel, in acest moment singurul cu acces la structura creata este insusi creatorul, insa acesta poate acorda acces cu diferite roluri (membru, editor sau admin) si altor utilizatori, adaungandu-se astfel keyspace-ul si in colectia acestora. Prin aceste facilitati, un membru poate sa-si creeze propria “comunitate” care sa interactioneze cu aceleasi date, chiar si in acelasi timp.

Cu toate ca mai multi useri pot accesa in acelasi timp aceleasi date, Cassandra Interface ofera securitate si in acest sens. Pentru fiecare interactiune cu baza de date, se verifica in primul rand daca utilizatorul este conectat la keyspace-ul pentru care se face actiunea, apoi daca are dreptul de a o executa (in functie de rolul pe care il detine in keyspace-ul respectiv). Conexiunea este stabilita dupa cum am precizat si mai sus, sub forma unei sesiuni, care va expira in lipsa inactivitatii, astfel un utilizator nu poate fi conectat in acelasi timp la mai multe keyspace-uri.

Alta caracteristica majora a securitatii bazei de date o reprezinta faptul ca toate interactiunile de editare sunt salvate in log-ul keyspace-ului. De asemenea, unele interactiuni creeaza un link de backup pentru modificarea facuta, din care se pot downloada, sub format JSON, datele de dinainte de modificare, obtinand astfel o siguranta continua. Pe langa aceste backup-uri dinamice, se pot exporta si manual datele si logurile printr-un singur click.

Prin toate aceste caracteristici, cat si cele descrise la capitolul anterior, aplicatia Cassandra Interface isi castiga securitatea si controlul asupra actiunilor factorilor externi.

Client-Side

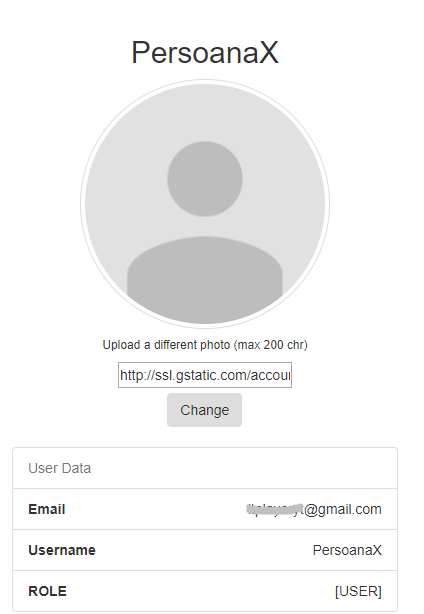
In aceasta parte vom vorbi despre facilitatile de care dispune aplicatia din punct de vedere al unui utilizator, fie el cu rol de membru, fie admin, utilizator neautentificat neavand acces decat la pagina de Home de unde se poate loga sau inregistra, si bineinteles, de a citi descrierea aplicatiei.

Am prezentat mai sus sistemul de autentificare si de securitate, care sunt valabile atat pentru utlizatorul normal cat si pentru admin. Mai departe vom elabora tot ceea ce aplicatia poate oferi.

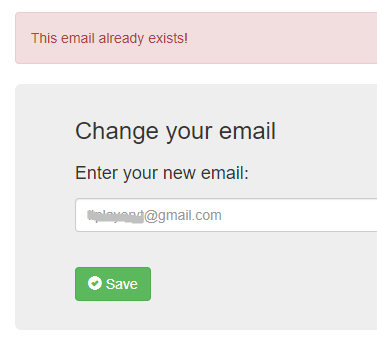
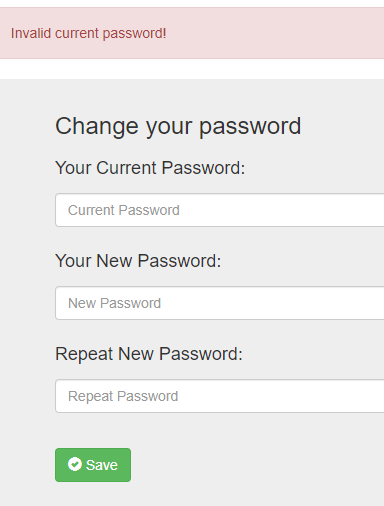
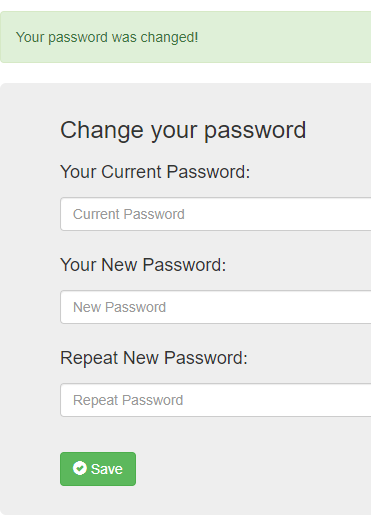
Sa presupunem ca o persoana (pot fi si mai multe), pe care o sa o notam arbitrar cu PersoanaX, are nevoie de o baza de date pentru a stoca datele afacerii sale. Citind despre diferite sisteme de gestiune, ajunge la concluzia ca o baza de date NoSQL i-ar satisface nevoile, astfel ca ajunge fata in fata cu aplicata Cassandra Interface. Dupa ce se intregistreaza, confirma adresa de email si se logheaza, PersoanaX are la dispozitie mai multe optiuni, insa alege sa-si consulte pagina de profil accesand butonul cu Username-ul ei din meniul aplicatiei (in partea dreapta, langa Logout).



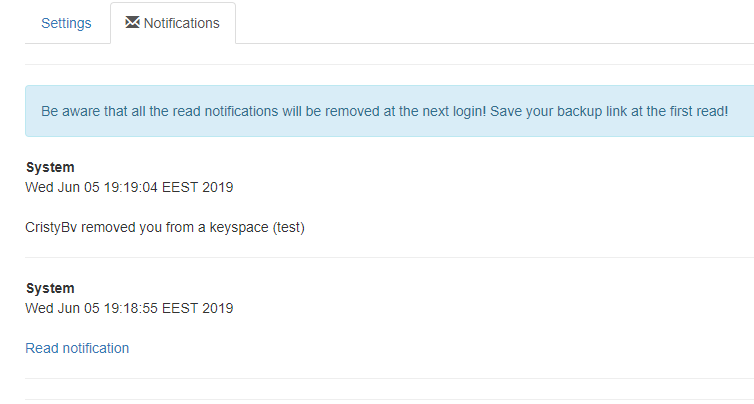
Ajuns pe pagina de Profil, PersoanaX observa ca sunt afisate cateva din datele personale si are la dispozitie mai multe setari asupra contului sau. Pentru inceput decide sa-si schimbe avatar-ul, inserand un URL al unei fotografii (URL ce nu trebuie sa depaseasca o anumita lungime), si apasa pe buton Change, care, folosit o cerere Ajax, transmite catre server (ProfileController) URL-ul si actualizeaza atat utilizatorul autentificat in context-ul Spring, cat si in baza de date.



Apoi, poate decide de exemplu sa-si schimbe email-ul sau parola, primind mesaje reprezentative daca operatiunea a avut succes sau nu.

De asemenea, tot in pagina de profil sunt afisate notificarile PersoaneiX (daca are). Acestea sunt transmise de catre Sistem cu ocazia anumitor evenimente cum ar fi: adaugarea utilizatorului intr-un keyspace, stergerea utilizatorului din keyspace sau stergerea keyspace-ului din care face parte. Pentru eficienta, aceste notificari, dupa cititre, vor fi sterse la urmatoroarea logare. Daca PersoanaX are notificari necitite, in dreptul Username-ului sau din meniu, va aparea un icon cu un plic pentru a-l anunta despre acest lucru.

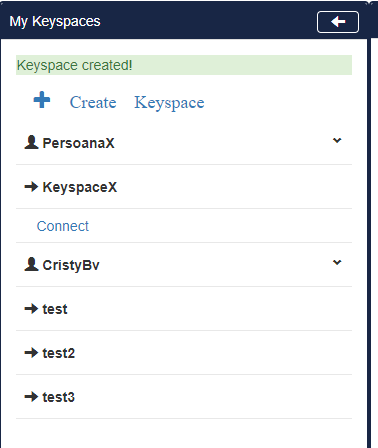
 

Acum ca PersoanaX si-a actualizat profilul, doreste sa inceapa sa foloseasca baza de date si sa-si invite colegii (daca este cazul) sa aiba acces la aceasta, asa ca da click pe butonul “My Database” din meniul aplicatiei.



Sa presupunem ca PersoanaX mai are cunostinte care folosesc aplicatia si a fost adaugat in trei keyspace-uri de catre acestia: primul are numele de “test” si aici X are rol de membru, apoi “test2” cu rol de editor si “test3” cu rol de admin. Cu toate acestea, inca nu doreste sa interactioneze cu acestea si decide sa-si creeze propriul keyspace, acesta fiind motivul pentru care foloseste aplicatia.

In partea stanga ii este prezentat tab-ul cu keyspace-uri (“My keyspaces”), care momentan contine butonul de creare si cele 3 keyspace-uri la care are acces. Astfel, da click pe “Create Keyspace” si ii apare modal-ul prezentat mai sus (la Securitate), unde isi seteaza datele pentru keyspace si il creeaza.



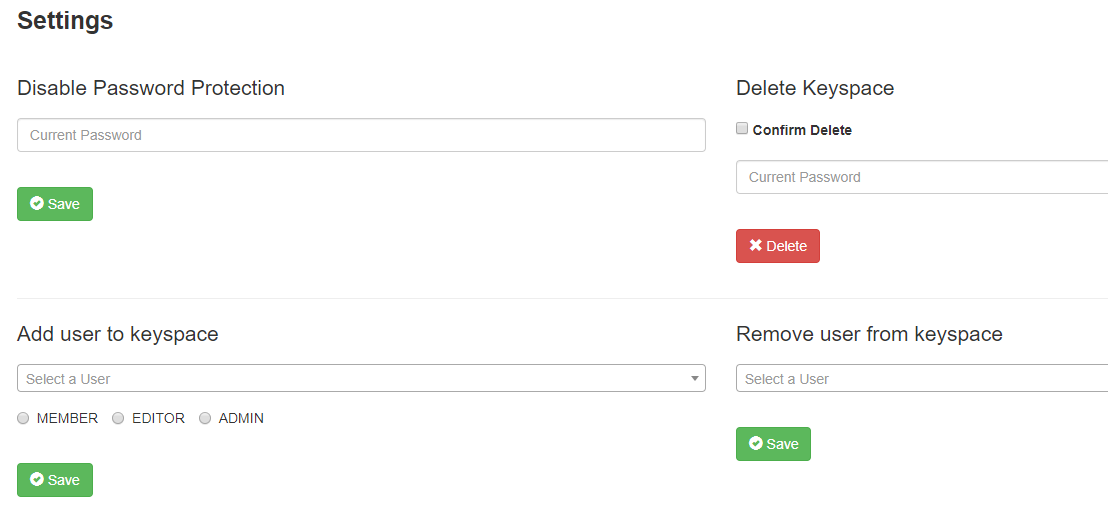
In poza de mai sus se poate observa mesajul de succes al crearii keyspace-ului cu numele “KeyspaceX”, creat de “PersoanaX”. De asemenea cele trei keyspace-uri de mai jos la care are acces, au ca si creator pe utilizatorul cu username-ul “CristyBv”.

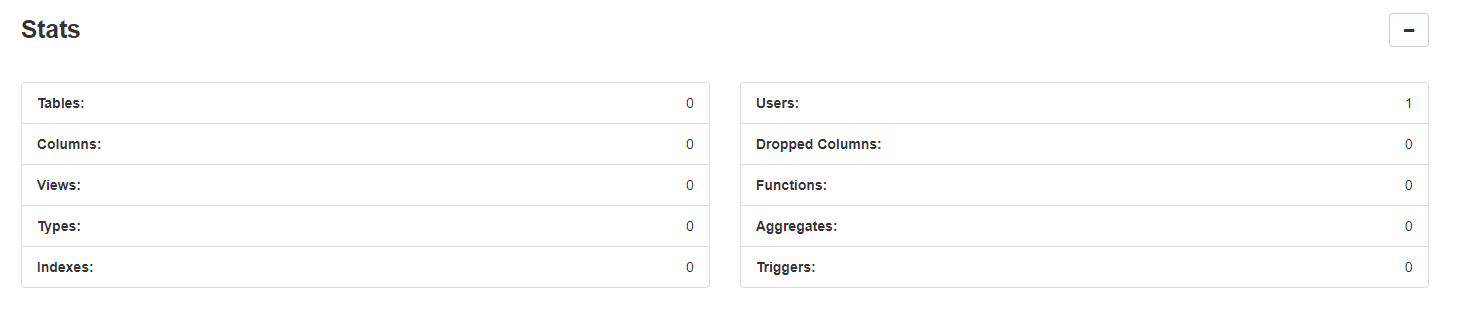
Mai departe, PersoanaX se conecteaza la keyspace prin butonul de Connect (folosind parola daca este cazul) si astfel, intern, se adauga la sesiunea curenta ca fiind conectat la KeyspaceX. Odata conectat, in partea libera a paginii apare un meniu reprezentativ pentru user-ul cu rol de creator si este redirectionat pe prima pagina a meniului, aceea de Manage.

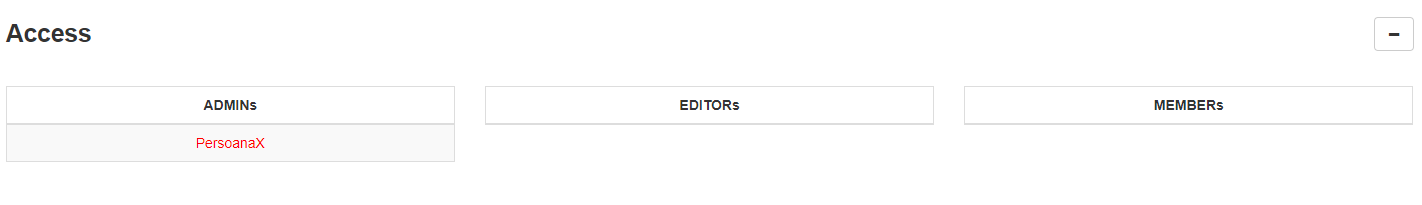


In aceasta pagina, PersoanaX poate vizualiza si modifica setarile keyspace-ului sau, il poate sterge, poate activa/dezactiva protectia prin parola, poate adauga/sterge membrii (folosind cautare Live prin cereri Ajax prin plugin-ul Select2 – va fi descris mai incolo). De asemenea, tot aici apar informatii utile despre content-ul keyspace-ului (numarul de tabele, coloane, etc), cat si tabele cu userii adaugati si rolurile lor.





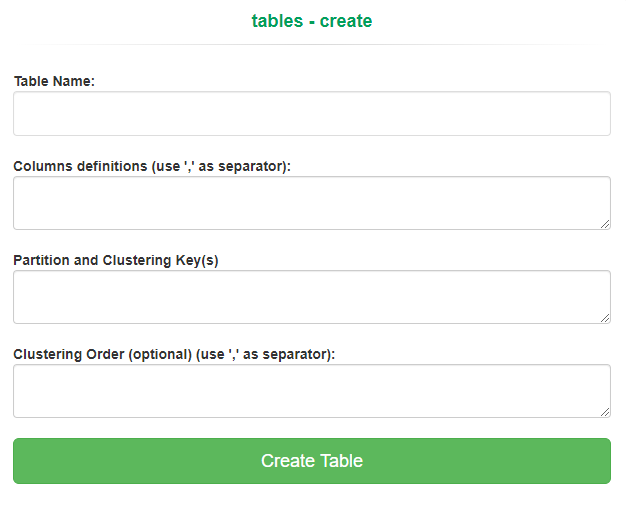


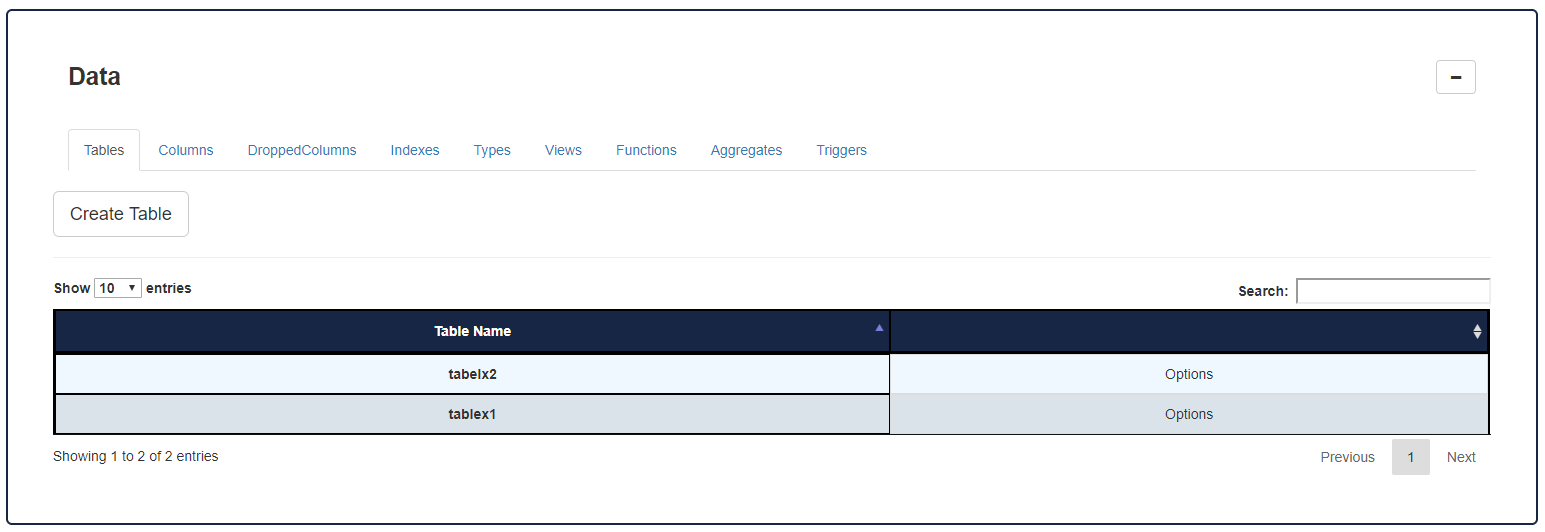


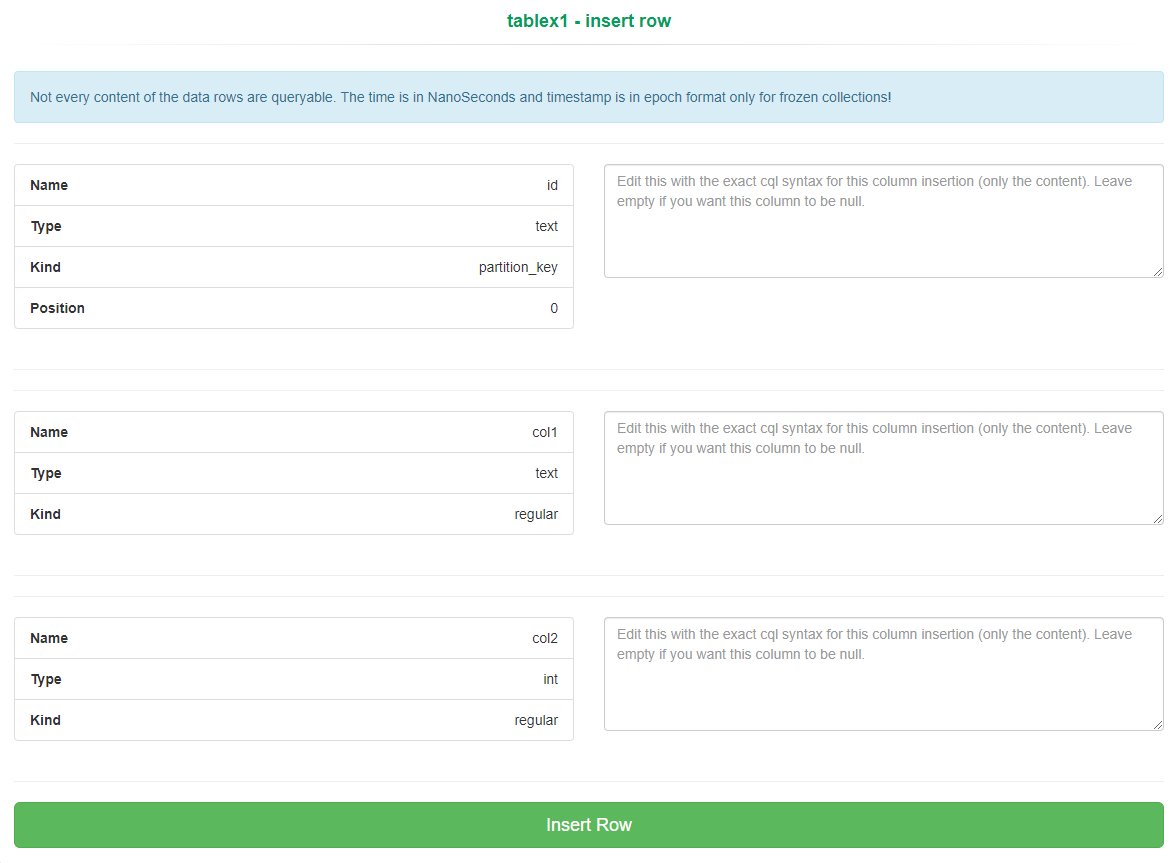
Dupa ce verifica sau editeaza aceste informatii, PersoanaX cerceteaza mai departe aplicatia si da click pe a doua pagina din meniu si anume View/Edit. Dand click pe View/Edit, intern, i se adauga in sesiunea curenta faptul ca pagina aleasa este aceasta, astfel, chiar daca paraseste baza de date, cand se va intoarce, aplicatia va stii sa afiseze aceasta pagina.

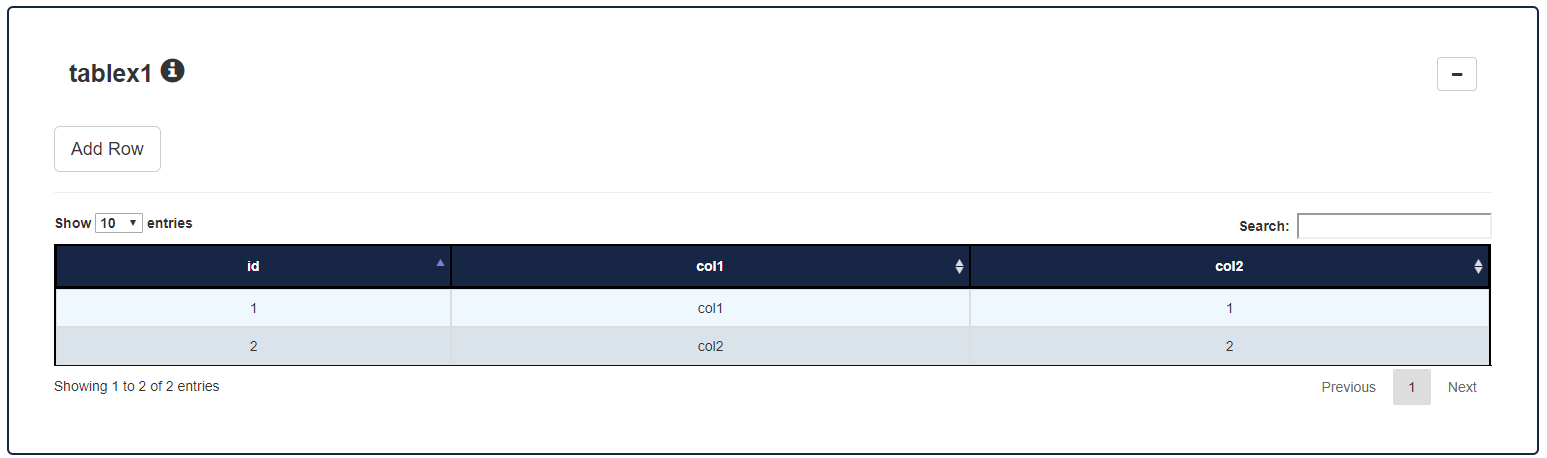
Pagina de View/Edit este impartita in 2 corpuri. Cel de sus (Data) unde sunt afisate toate tabelele / coloanele / indecsii / type-urile / functiile / etc, fiecare cu proprietatile lor, putand fi adaugate / editate sau sterse unde este cazul. In corpul de jos (Content) vor fi afisate datele dintr-un tabel / view, atunci cand se da click pe unul (de asemenea, alegerile sunt memorate in sesiune), si la fel ca mai sus, se pot adauga, edita sau sterse linii in tabel.

PersoanaX incepe prin a crea cateva tabele si insera cateva date pentru a testa aplicatia.





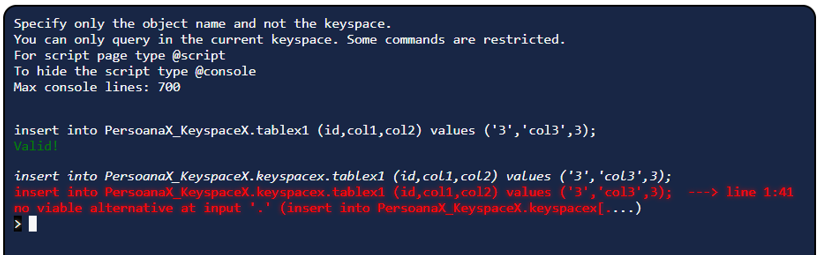




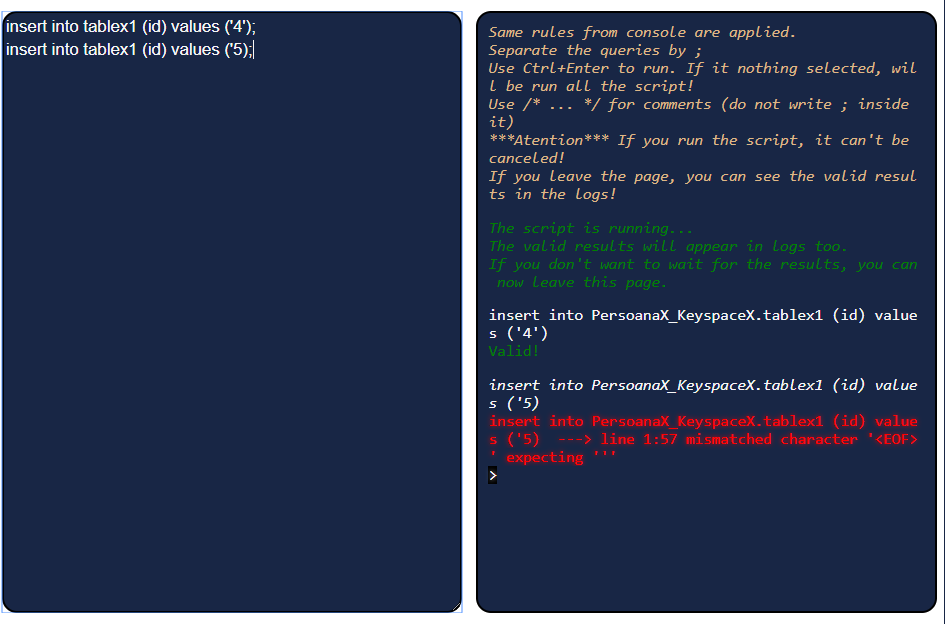
In imaginile de mai sus se poate observa cum PersoanaX, da click pe Create Table, introduce datele corespunzatoare, creand cele 2 tabele, care apar in tabelul de mai jos. Apoi, dand click pe unul dintre acestea, respectivul va aparea in corpul de jos, de unde X va insera cele doua linii. Este de precizat ca cele doua tabele (create cu plugin-ul DataTable – va fi descris mai incolo) primesc datele diferit. Cel de sus, avand date relativ putine, vor fi incarcate toate in partea de client-side, in timp ce tabelul de jos, putand avea sute de mii de date, isi preia datele (filtrate sau nu) din server-side, imbunatatind astfel considerabil eficienta aplicatiei.

Mai departe, platforma ofera mai multe setari. In afara de paginare, sortare, cautare, facilitati oferite de plugin, PersoanaX poate redimensiunea coloanele tabelului prin scroll up/down, poate ascunde coloane prin dreapta click pe acestea si il poate reseta prin click pe titlul acestuia. Pentru a edita o linie, X trebuie doar sa dea dublu-click pe aceasta si ii va aparea un modal cu datele respective pe care le poate upadata sau sterge.

Toate aceste operatiuni se repeta mai mult sau mai putin pentru toate structurile din Cassandra (tabele / coloane / functii / etc) in pagina de View/Edit. Mai departe, PersoanaX ajunge la pagina de Console/Script, unde este activat plugin-ul de terminal (va fi descris mai incolo), afisandu-l sub forma unei console in care X poate introduce aproape totate tipurile de query existente in Cassandra (create / alter / drop / select / insert / update / delete / …). Aplicatia fiind securizata, nu va valida niciun query destinat altui keyspace in afara de cel curent, astfel, query-urile nu trebuie sa contina si numele keyspace-ului, fiind adaugat automat intern.



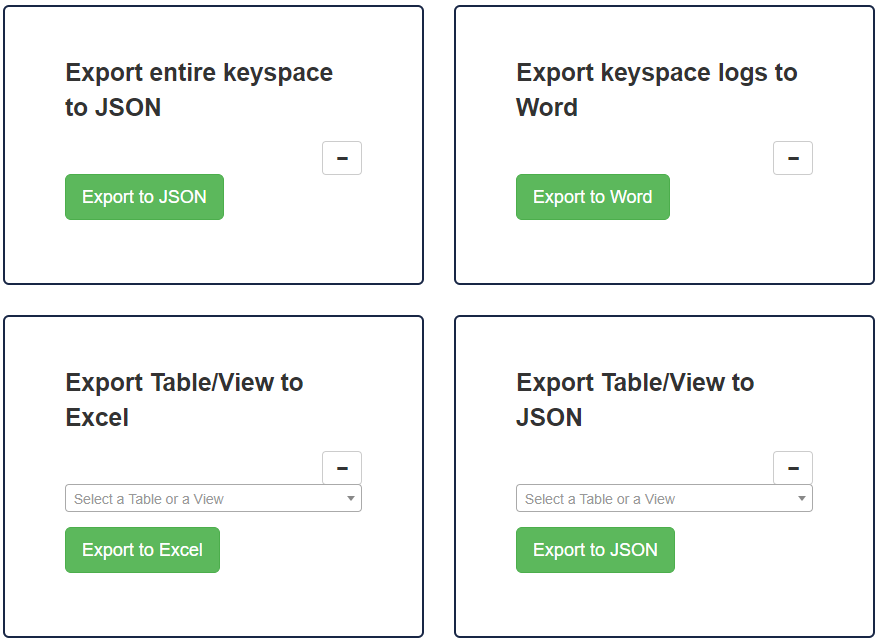
Pe langa facilitatea de consola, prin tastarea comenzii @script, va aparea inca un corp in care se pot introduce mai multe query-uri odata si rulate (prin Ctrl+Enter, daca nu este nimic selectat, va fi rulat tot scriptul, altfel doar ceea ce este selectat).



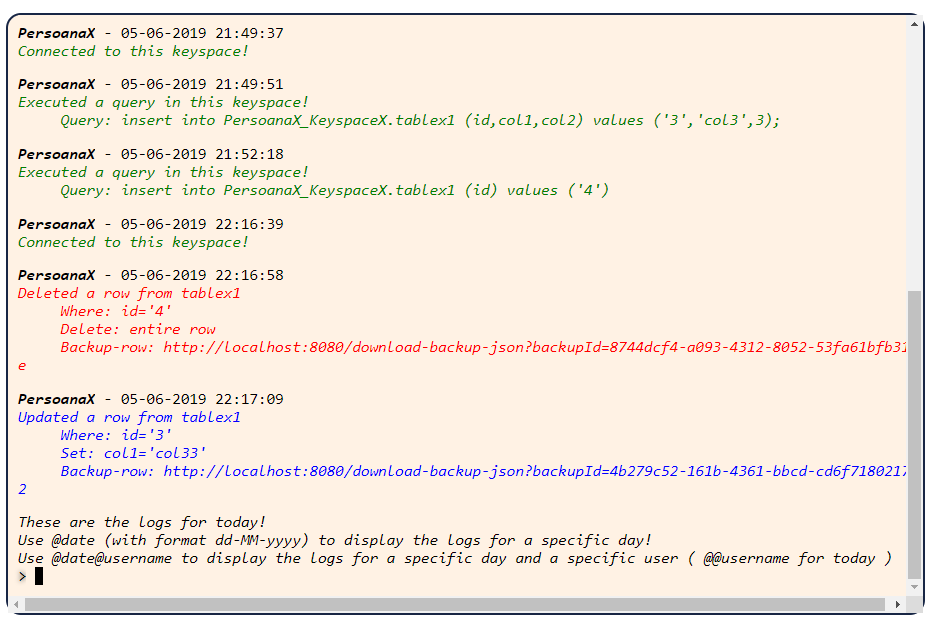
Acest mecanism este posibil prin cereri Ajax catre server cu respectivele query-uri, care se detecteaza, valideaza si executa, returnand rezultatul in consecinta. Astfel, odata ce scriptul a fost activat, cererea Ajax este trimisa, iar PersoanaX daca nu doreste sa astepte rezultatele (in caz ca sunt foarte multe query-uri), poate parasi pagina, fiindca oricum editarile valide vor aparea de asemenea si in log-ul keyspace-ului.

Ca si aproape celelalte facilitati, atat alegerile cat si continutul din console/script, este salvat in sesiune pentru ca PersoanaX sa nu piarda continut din greseala. In cazul unei comenzi de select, in care datele pot fi nenumarate, consola afiseaza numai cateva, lasand posibiltatea tastarii comenzii @more pentru a incarca mai multe dintre acestea, evitand astfel blocarea paginii.

Mai departe, X ajunge la pagina de Export, in care are posibilitatea atat de a-si exporta tabelele sau view-urile ca Excel (folosind libraria ApachePOI) dar si ca JSON (folosind libraria Jackson), cat si posibilitatea de exportare a intregului keyspace ca JSON sau a log-ului ca fisier de tip Word (folosind de asemenea ApachePOI).

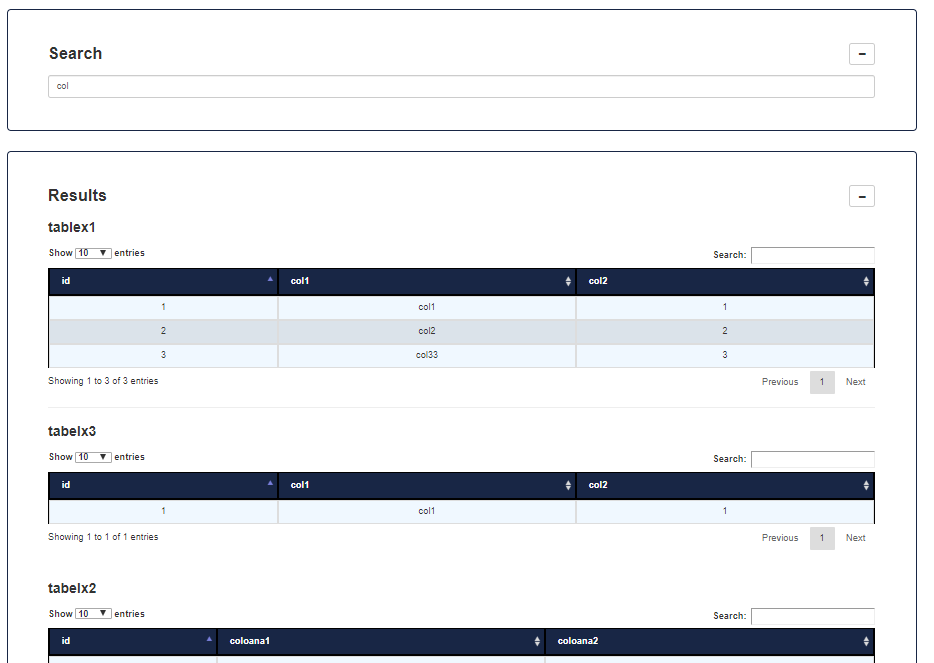


Urmeaza pagina de Log, in care tot sub forma unei console sunt afisate log-urile keyspace-ului pentru ziua curenta. Pentru a vedea alte log-uri, PersoanaX trebuie sa urmeze comenzile afisate in consola.



Dupa cum se poate vedea, log-urile pentru conectare / creare / inserare sunt reprezentate de culoarea verde, cele de alter / update de albastru, iar cele pentru deconectare / drop / delete de rosu. Mai mult decat atat, dupa cum am precizat si mai sus, editarile / stergerile facute in pagina de View/Edit, genereaza automat si un fisier JSON de backup, iar link-ul acestuia apare in loguri.

Ultima, este pagina de Search, unde PersoanaX poate cauta in toate tabelele din keyspace-ul lui. Rezultatele sunt grupate dupa aceste tabele, iar daca sunt prea multe rezultate, se afiseaza numai cateva, si se sugereaza sa se restranga criteriul de cautare.



Acum ca PersoanaX a parcurs si a testat functionalitatile aplicatiei pentru keyspace-ul sau, urmeaza sa se conecteze si la celelalte keyspace-uri in care a fost adaugat. Facand asta, el observa urmatoarele:

Rolul de membru nu-i ofera decat informatii despre keyspace si acces de vizualizare, adica la pagina de View, insa tot ceea ce este legat de editare / stergere nu mai apare, la pagina de Export, unde poate exporta numai tabele sau view-uri si in cele din urma, are acces la Search.

Rolul de editor, fata de membru, ii ofera si posibilitatea de a vizualiza, pe langa informatiile keyspace-ului, si statisticile acestuia cat si utilizatorii adaugati. Fiind editor, are acces de editare / stergere in cadrul paginii de View/Edit, poate folosi de asemenea pagina de Console/Script. La Export si Search este la fel ca la member.

Rolul de admin ii ofera toate drepturile creatorului, in afara de: posibilitatea de a modifica setarile keyspace-ului, de a activa / dezativa protectia cu parola, si evident, de a sterge keyspace-ul.

Astfel, PersoanaX cunoaste aplicatia si poate incepe sa-si invite echipa de afaceri si sa-si populeze baza de date personala dupa bunul plac.

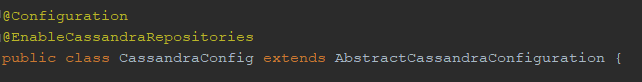
Diferenta dintre un utilizator normal al aplicatiei si unul cu rol de ADMIN, este aceea ca adminul are la dispozitie inca o pagina (disponibila in meniul aplicatiei), numita Console. Aici apare acelasi terminal ca in pagina din baza de date Console/Script (dar fara partea de script), insa nicio restrictie nu mai este aplicata, query-urile putand contine orice keyspace din toata baza de date. Astfel, un admin poate face absolut orice modificare asupra aplicatiei prin intermediul bazei de date, inclusiv adaugarea altui admin, modificand in acest sens tabelul de useri din keyspace-ul admin.

Server-Side

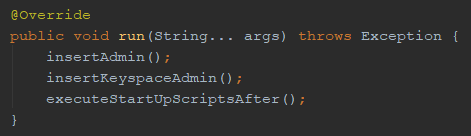
Dupa cum a fost prezentat si la sub-capitolul Structura, backend-ul cuprinde doua parti importante: cea de configurare si cea de web. Vor fi discutate pe rand.

Pentru inceput, tot mai sus am precizat ca aplicatia este dependenta de baza de date, dat fiind scopul acestia. Astfel, o parte important din configuratiile aduse aplicatiei sunt pentru Apache Cassandra. Datorita suportului adus de catre aceasta in aplicatiile Java, a fost folosita o clasa de configurare automata a bazei de date, in care au fost suprascrise anumite metode pentru adaptare. Acesta clasa se numeste AbstractCassandraConfiguration si a fost importat prin dependinte Maven.

Odata ce baza de date Apache Cassandra este pornita pe server, clasa CassandraConfig, care extinde AbstractCassandraConfiguration (cea precizata mai sus), auto-configureaza aplicatia, conectand-o cu instanta bazei de date pe care o detecteaza pe server cu ajutorul metodelor suprascrise cu datele de conectare preluate din fisirul de proprietati al aplicatiei. Mai mult decat atat, unele metode vor prelua din anumite fisiere de pe server (startUpScripts, shutDownScripts), scripturi predefinite care sa se execute la pornirea, respectiv oprirea server-ului. Aceste scripturi constau in crearea / stergerea unor materialized-view-uri pentru tabelele definite in aplicatie, spre a putea prelua date din acestea dupa un alt criteriu in afara de cheia primara (acest mecanism este descris la capitolul Tehnologii folosite).



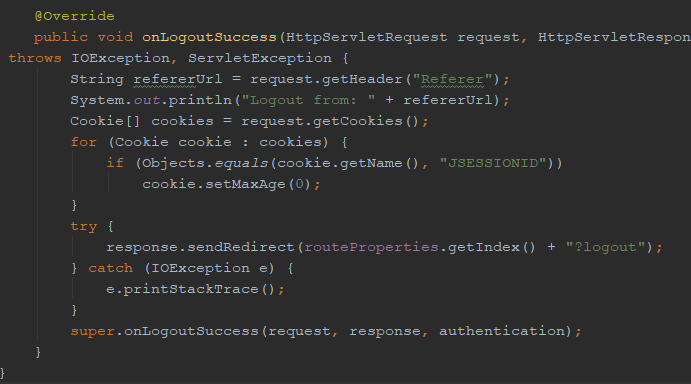
Trebuie precizat insa ca, in Cassandra, un tabel nu poate fi sters sau recreat daca materialized-view-ul care depinde de el inca exista; acest lucru este valabil si invers, un materialized-view neputand fii creat daca tabelul de baza nu exista. Astfel, din cauza acestui lucru, atat StartUpScrips cat si ShutDownScripts fac acelasi lucru si anume sa stearga toate materliazed-view-urile care depind de tabelele ce vor fi recreate la pornirea server-ului Spring, astfel eliminand orice sansa de a aparea vreo eroare la pornire. Apoi, dupa ce AbstractCassandraConfiguration isi face treaba, si creeaza tot ceea ce trebuie creat, intr-o clasa numita CommandLineAppStartRunner care implementeaza CommandRunner, permite prin metoda “run” a se rula cod dupa pornirea cu succes a server-ului Spring Boot, putand astfel aici rula scripturile de creare dintr-un alt fisier numit “startUpScripsAfter”. Tot in acesta metoda sunt inserate conturile default si keyspace-ul de admin, in care se afla toate datele de constructie a aplicatiei (useri / keyspace-uri / etc).



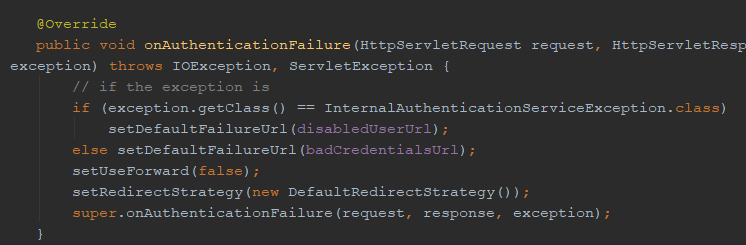
Mare parte din configurarea framework-ului Spring Security a fost descrisa anterior, insa mai sunt de precizat cateva amanunte.

Ca si Apache Cassandra, si SpringSecurity a trebuit importat prin Maven in fisierul de configurare pom.xml al aplicatiei. Cu acest import a fost posibila crearea clasei SecurityConfig care extinde clasa de configurare WebSecurityConfigurationAdapter, permitand astfel suprascrierea anumitor proprietati. In aceste setari descrise si anterior, este de spus ca au fost intoduse clase de tip “handle” pentru anumite evenimente.

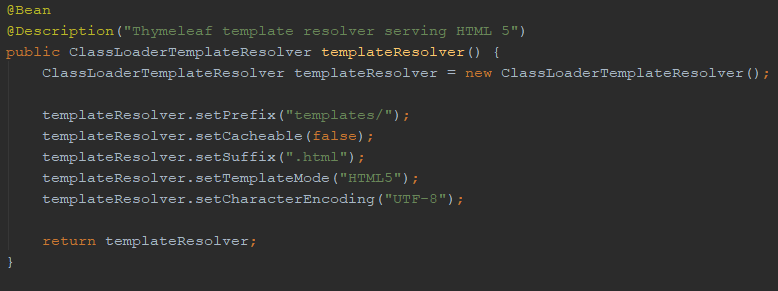
Unul din aceste handlere este “CustomLogoutHandler”, care se apeleaza atunci cand utilizatorul doreste sa se delogheze. In mare parte, aici sunt afisate mesaje in consola server-ului, se sterg cookie-uri si se redirectioneaza catre pagina de Home, urmand ca Spring Security sa-si continue protocolul.



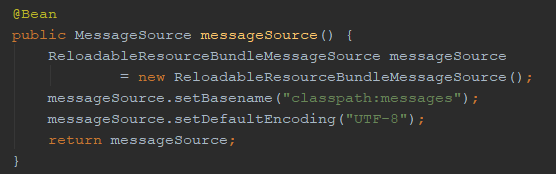
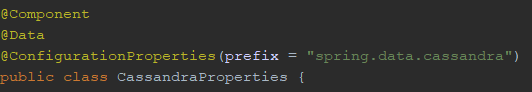
Un alt handler important este “CustomAuthenticationFailureHandler”, care se apeleaza atunci cand este anuntata o eroare la autentificarea user-ului in aplicatie. Aici se verifica ce tip de eroare este si se stabileste mesajul de eroare potrivit.



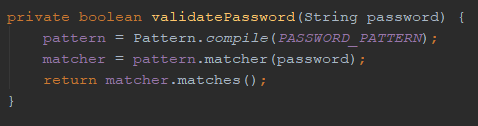
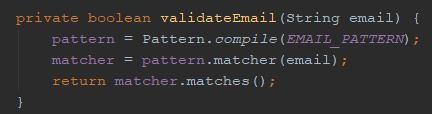
O configurare importanta o constituie cea de setare a comunicarii cu frontend-ul. Clasa WebMvcConfig implementeaza clasa predefinita pentru aplicatii web (importata tot prin Maven), WebMvcConfigurer, care permite setarea “engine”-ului de compilare a template-urilor. Astfel aici este configurat Thymeleaf-ul, setand locatia (prefixul), extensia (sufixul) si formatul documentelor de html pe care sa le proceseze, cat si dialectele folosite (descrise mai sus).

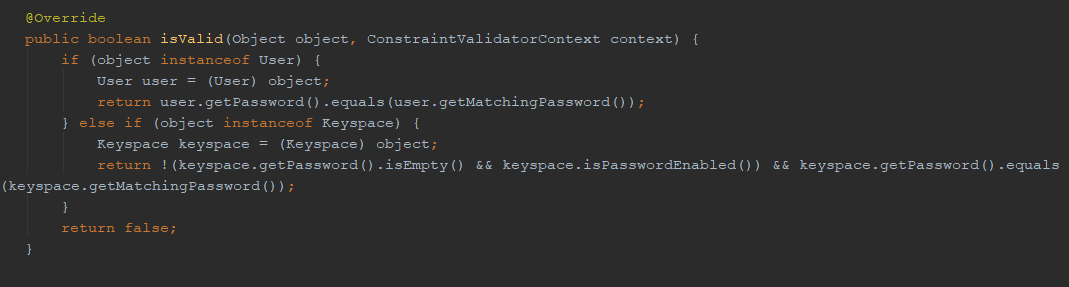


Tot la partea de configurare sunt adaugate clasele de proprietati: “CassandraProperties”, “KeyspaceProperties”, “QueryProperties”, “RouteProperties”, prin care se preiau setari din fisierul “application.properties” (fisier de proprietati specific pentru Spring). De asemenea, clasa “MessagesGetter”, care face acelasi lucru pentru fisierul de message “messages.properties” si clasa “ResourceGetter”, care preia continutul altor fisiere (cum ar fi cele descrise mai sus pentru scripturi).



In final, pentru partea de configurare mai intra si clasele de validari, cum ar fi “EmaiValidator”, “PasswordMachesValidator” si “PasswordPatternValidator”, care, setate pe un anumit camp (sau chiar clasa in cazul PasswordMachesValidator), ajuta la validarea continutului pentru acel camp / clasa. Astfel, validarea pentru email se face dupa un anumit pattern specific email-urilor, validarea pentru parole identice compara cele doua parole, iar cea pentru pattern-ul parolei, de asemenea foloseste un regex pentru verificarea parolei.



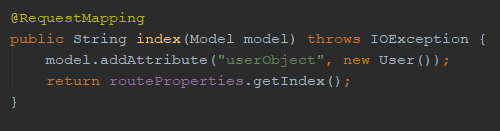


Urmeaza partea de web, care consta in mai multe controllere, modele, servicii, evenimente si alte caracteristici acestei comunicari intre frontend si backend (REST).

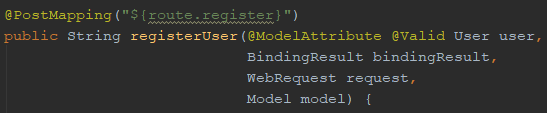
Aplicatia Cassandra Interface, are 9 clase ce indeplinesc rolul de controller. Folosind Thymeleaf, controllerul poate returna un string cu numele view-ului care trebuie afisat. Pentru stilizare si eficienta, numele respectiv este preluat din fisierul de proprietati.

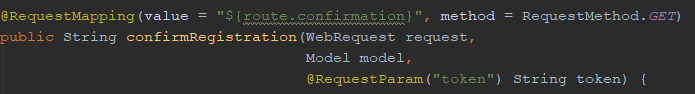
De asemenea este important de precizat ca, fiind o aplicatie in care cererile sunt aproape intotdeauna dinamice (tabele, tipuri, functii, fisiere, etc create dinamic de catre useri), iar modelele sunt denormalizate (datele cu interes comun in principiu sunt toate intr-un singur tabel, dat fiind ca este o baza de date NoSQL, adica nerelationara) , aplicatia primeste aceste cereri numai sub forma de GET si POST (in general cerearea GET este cand se preiau date din baza de date, iar POST cand se modifica), nefolosind astfel cereri de tip PUT sau DELETE. In consecinta, controllere accepta numai astfel de cereri (GetMapping si PostMapping – adnotari folosite pentru framework-ul Spring pentru a stii ce fel de cerere sa intercepteze controllerul).

Primul este HomeController si are ca scop prelucrarea request-urilor pentru pagina de Home. In esenta, acesta trebuie doar sa creeze obietul User pe care sa-l transmita in view, ca Thymeleaf sa-l poata folosi pentru formularul de register.



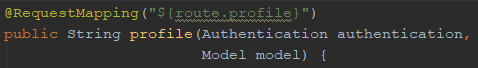
Urmatoarele trei sunt legate de inregistrarea in aplicatie a unui utilizator. Primul din cele trei este RegisterController care primeste datele din formularul de inregistrare, le valideaza si returneaza rezultatul. Apoi urmeaza ConfirmationController care este apelat atunci cand utilizatorul da click pe link-ul trimis pe email, verificand astfel token-ul de confirmare si validandu-l sau respingandu-l daca este cazul. Ultimul este RecoveryController, care primeste datele din formularul de recovery, testeaza ce tip de cerere este (pentru resestare parola sau pentru retrimitere email de activare) si actioneaza in consecinta.







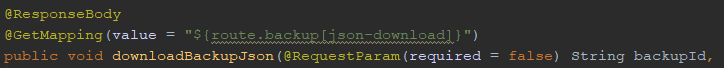
Urmatorul este ProfileController si are ca scop prlucrarea request-urilor pentru pagina de Profil. Ruta principala, care returneaza pagina de profil are sarcina de a pregati notificarile utilizatorului (daca exista), sortandu-le descrescator dupa data si anuntarea daca sunt sau nu notificari necitite. Apoi mai sunt rutele care sunt apelate la anumite evenimente: schimbarea avatarului (care primeste o cerere Ajax si returneaza un obiect in format JSON), citirea notificatilor, schimbarea parolei si a email-ului, email care si de data aceasta trebuie confirmat.



Mai departe vorbim despre SearchController, care se ocupa de cererile Ajax provenite de la cautarile Live din cadrul aplicatiei. Se poate observa aici adnotarea diferita a controllerului, fiind @RestController in loc de @Controller ca la celalalte, diferenta fiind ca un RestController va intoarce un “body” (de obicei string JSON) in loc de un view, rezultat acceptat de catre cererile Ajax. Cautarea Live este eficienta datorita faptului ca rezultatele apar in timp real pe masura ce parametrul de cautare este scris, elementele astfel (in numar mare si nefiltrate) nefiind afisate in view de la inceput, economisindu-se astfel memorie si spatiu. Astfel, acest controller se ocupa de cautarile Live pentru useri, tabele, view-uri, coloane si functii, pregatind datele pentru ca plugin-ul Select2 sa le poata primi si afisa in frontend (vom discuta mai incolo despre acest sistem).



BackupController se ocupa de descarcarea unui anumit backup (dupa id) sub forma unui fisier JSON (backup-ul are deja stocate datele sub forma unui JSON, acest controller doar trebuind sa le copieze in raspuns-ul catre frontend, care este transformat intr-un raspund de tip fisier). Controller-ul cauta backup-ul dupa id in baza de date, il gaseste daca exista si actioneaza in consecinta. Raspunsul returnat fiind un fisier si nu un view, in loc de adnotarea @RestController asupra clasei, s-a ales folosirea adnotarii @ResponseBody, care semnifica acelasi lucru insa nu pe toata clasa ci doar pe functia respectiva.



Urmatorul este AdminConsoleController care are sarcina de a gestiona cererile legate de pagina Console, disponibila numai adminilor. In principiu acest Controller verifica autenticitatea persoanei care il acceseaza (sa aiba drept de acces), iar la trimiterea unei comenzi (query) printr-o cerere Ajax, o proceseaza si returneaza rezultatul care va fi apoi afisat in consola.

Ultimul si cel mai complex este MyDatabaseController care se ocupa de toate cererile care tin de paginile din My Database descrise la Client-Side. Pentru inceput putem spune ca aproape toate metodele sunt pe baza sesiunii utilizatorului. Odata ce acesta se conecteaza la un keyspace, mereu se va testa daca acel keyspace este inca activ si bineinteles daca are acces. Astfel, acest controller poate primii cereri pentru:

* Conectare, deconectare, editare, stergere keyspace.
* Adaugare / stergere membru keyspace
* Crearea, stergerea, alterarea unei structuri.
* Inserarea, editarea, stergerea datelor dintr-o structura.
* Selectarea datelor dintr-o structura si pregatirea lor pentru afisare (in functie de tipul de date).
* Executarea query-urilor dinamice din consola / script, detectand tipul acestora si actionand in consecinta.
* Exportarea unei structuri.
* Crearea de log-uri pentru fiecare modificare a bazei de date.
* Schimbare panel (pagina) din MyDatabase (actualizare sesiune).
* Schimbare continut in consola sau script (cerere ajax).
* Cautare in baza de date.

Atunci cand utilizatorul preia date din baza de date sau le vizualizeaza in pagina de View, acestea trebuie procesate inainte, deoarece Cassandra, avand numeroase tipuri de date si colectii, integrarea cu Java ingreuneaza acest proces mai ales ca datele sunt dinamice. Astfel, tot in acest controller a fost simulata o interpretare a datelor in functie de tipul lor, iar in cazul colectiilor s-a folosit o functie recursiva pentru a altera primitivele din acestea. Pentru cazurile tipulor frozen, care pot fi de orice tip (creat sau nu de utilizator), aplicatia inca nu ofera suport vizual la fel de bun, insa in viitor se va implementa si acest lucru. Din acest motiv nici importul datelor nu este posibil momentan.

De asemenea, fiind o aplicatie la care mai multi utilizatori pot executa comenzi in acelasi timp, datele sunt re-actualizate inainte de fiecare actiune pentru a nu exista conflicte. De asemenea, sunt afisate mesaje informative (preluate din fisierul de mesaje) pentru fiecare actiune, astfel utilizatorul va stii mereu in ce situatie se afla. Astfel, acest controller se ocupa de toate aceste caracteristici, facilitand astfel comunicarea dintre utilizator si baza de date.

Acum urmeaza descrierea modelelor aplicatiei. Totul fiind mai mult dinamic, nu exista decat 3 modele si anume User, Keyspace si Backup. Aceste modele la randul lor au anumite campuri de tip “frozen”, fiind create 4 tipuri (UDT) pentru ele. Sunt facute in asa fel incat sa contina destul de multe informatii intr-un singur “tabel”, fiind o baza de date NoSQL, relatiile nu exista, iar tabelel sunt eficiente daca sunt denormalizate. In continuare urmeaza prezentarea lor:

Modelul User are 13 campuri:

* Id – string, cheiea primara in care se va depozita un id de forma UUID (universally unique identifier).
* Email – string, validat cu validatorul de email descris mai sus.
* Username – string, validat cu pattern si indexat in baza de date, fiind un camp foarte folosit.
* Password si MatchingPassword – string-uri, trebuie sa fie identice (se valideaza intreaga clasa), insa in baza de date este trecut numai campul Password.
* Roles – lista de stringuri, contine rolurile de care dispune un user in aplicatie (user sau admin).
* RegisterDate – tip date, data de inregistrare.
* Enabled – boolean, true daca user-ul a confirmat email-ul, false in caz contrar.
* Token – string, retine token-ul care va fi comparat cu cel trimis pe email pentru confirmarea acestuia.
* ExpiryDate – tip date, retine data la care token-ul de confirmare va expira.
* EmailToken – string, retine token-ul care fi comparat cu cel trimis pe email pentru schimbarea acestuia.
* Avatar – string, retine url-ul avatarului utilizatorului.
* Notifications – lista de UserNotification (UDT), retine notificarile utilizatorului. UserNotification contine urmatoarele:
  + Author – string, retine autorul notificarii.
  + Date – tip data, contine data la care este trimisa notificarea.
  + Read – boolean, true daca notificarea a fost citita, false in caz contrar.
  + Content – string, contine continutul notificarii.
* Keyspaces – lista de UserKeyspace (UDT), retine keyspace-urile din care face parte userul. UserKeyspace contine urmatoarele:
  + Keyspace – acest camp nu apare in baza de date, dar exista pentru ca acest UDT este folosit pentru a retine in sesiunea utilizatorului atunci cand este conectat la un keyspace, iar in acest camp se va retine keyspace-ul (modelul) propriu-zis.
  + Name – string, numele keyspace-ului.
  + Access – string, accesul pe care il are userul in keyspace-ul respectiv.
  + CreatorName – string, numele creatorului keyspace-ului.

Modelul Keyspace are 9 campuri si este de precizat ca la crearea unui keyspace, pe langa ca este trecut in acest tabel, este si creat fizic in Cassandra.

* Id – string, cheiea primara in care se va depozita un id de forma UUID.
* Name – string, validat cu pattern, retine numele keyspace-ului care este format astfel: NumeCreator\_NumeKeyspace, astfel evitand conflictele de nume in baza de date.
* PasswordEnabled – boolean, true daca este protejat cu parola, false in caz contrar.
* Password si MatchingPassword – string-uri, trebuie sa fie identice (se valideaza intreaga clasa), insa in baza de date este trecut numai campul Password.
* CreationDate – tip data, retine data la care este creat keyspace-ul.
* ReplicationFactor – integer, minim 1 – maixm 8, retine factorul de replicare al keyspace-ului.
* DurableWrites – boolean, true daca keyspace-ul permite durable writes, false in caz contrar.
* Log – lista de KeyspaceLog (UDT), retine log-urile pentru keyspace-ul respectiv. KeyspaceLog contine urmatoarele:
  + Content – string, retine continutul logului.
  + Date – tip data, retine data la care a fost creat logul.
  + Username – string, retine userul care prin actiunile lui a creat logul respectiv.
  + Type – string, retine tipul logului, care poate fi de: creare, editare, stergere.
* Users – lista de KeyspaceUser (UDT), retine userii care apartin de keyspace-ul respectiv. KeyspaceUser contine urmatoarele:
  + Username – string, retine username-ul utilizatorului care are acces la acest keyspace.
  + Acces – string, retine accesul pe care il are utilizatorul cu acces la acest keyspace.

Modelul Backup are 4 campuri:

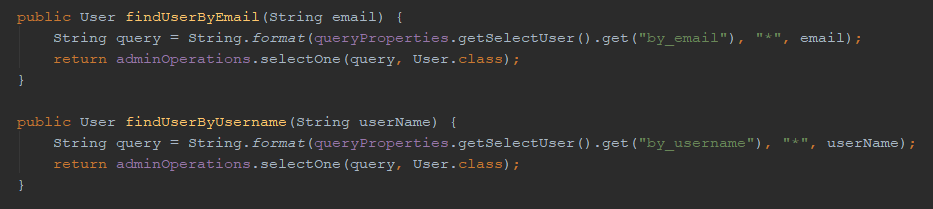
* Id - string, cheiea primara in care se va depozita un id de forma UUID.
* Content – string, retine continutul backup-ului, in general sub format JSON.
* Date – tip data, retine data la care a fost creat backup-ul.
* KeyspaceName – string, retine numele keyspace-ului in contextul caruia a fost creat backup-ul.

Aceste modele sunt adnotate in primul rand cu @Table, notatie care spune framework-ului sa creeze aceste tabele fizic in baza de date (fiind integrat cu Cassandra, acest lucru este posibil). Apoi, mai sunt adnotatii din cadrul librariei Lombok, care o sa fie discutata mai incolo).

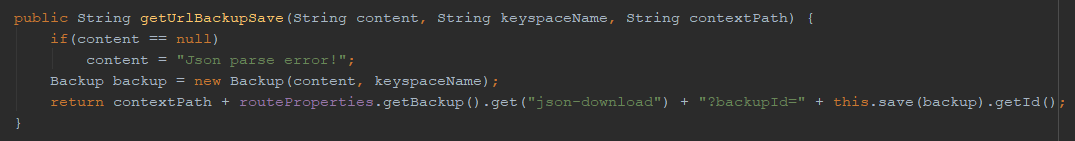
Mai departe vorbim despre clasele adontate cu @Service, notatie care spune framework-ului sa trateze aceste clase ca facand parte din contextul aplictiei si care vor fi folosite spre a oferi diferite servicii. In acest caz, acestea se vor ocupa de diferite tranzactii asupra bazei de date, folosind un “bean” creat la configurarea Cassandrei (descrisa anterior) si anume CassandraAdminOperations, care permite conexiunea directa cu baza de date, permitand executarea comenzilor si preluarea datelor. Aplicatia Cassandra Interface are 3 servicii.

Primul este UserService, care se ocupa de tranzactiile pentru tabelul modelului User, descris mai devreme. Aceste tranzactii pot fii de: adaugare, stergere, editare, activare si cautare dupa mai multe criterii.

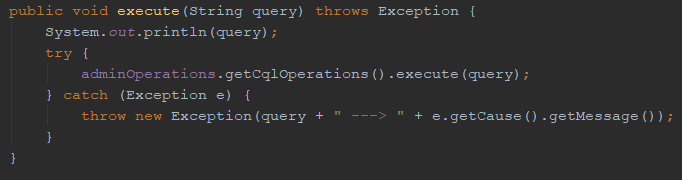




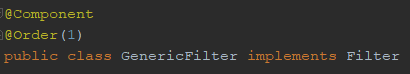
Al doilea este BackupService, care se ocupa de crearea, cautarea dupa id si formarea link-ului de download pentru un obiect de tip Backup.

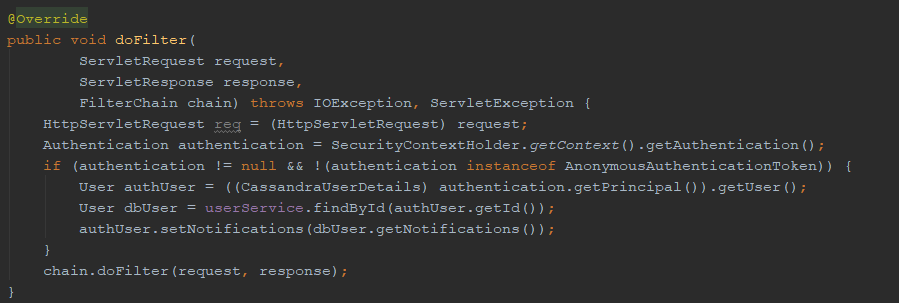


Ultimul este KeyspaceService, care serveste la mai multe tranzactii, nu numai celor legate de modelul Keyspace. In afara de tranzactiile pentru modelul Keyspace, tot de acest serviciu se foloseste aproape intreg MyDatabaseController-ul, facilitand crearea, editarea, stergerea si preluarea tuturor structurilor din baza de date create dinamic de catre utilizatori. Acestea se afla in acest serviciu datorita faptului ca utilizatorii depind si pot interactiona cu baza de date numai prin intermediul keyspace-ului la care sunt conectati.

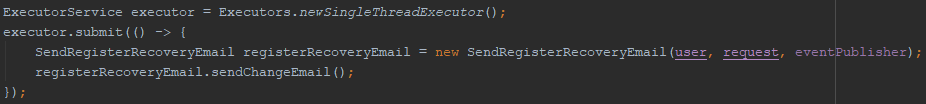


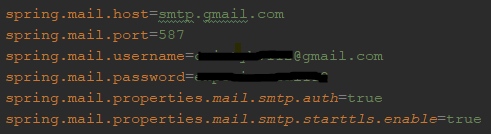
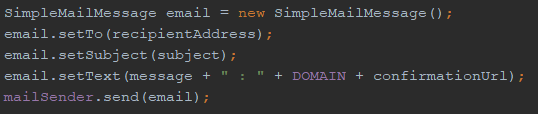
Tot pe partea de server in categoria web, aplicatia mai contine GenericFilter, o clasa care implementeaza clasa Filter si este adnotata cu @Component, spunand astfel framework-ului sa creeze un bean si sa foloseasca clasa acesta in scopul pentru care a fost defnitia (daca are un scop). In aplicatiile web, filterul are rolul de a se apela inaintea controller-ului, pentru a “filtra” informatiile din request. Pentru aplicatia Cassandra Interface, filterul este apelat inaintea oricarui Controller, verificand daca utilizatorul care a facut cererea este autentificat, iar in caz pozitiv, actualizeaza notificarile acestuia in contextul Spring. Acest sistem este important pentru a afisa in timp real utilizatorului notificarile primite, iar utilizarea filter-ului este esentiala fiind inutila actualizarea in fiecare controller separat.





Acum vorbim despre sistemul de trimitere al email-ului de confirmare / recuperare. Pentru ca aplicatia sa nu astepte pana la efectuarea acestuia, s-a folosit un Thread. Thread-ul permite executarea in paralel a unui alt proces. Pe langa acest lucru, a fost folosit sistemul de evenimente, care stabileste trimiterea email-ului ca un eveniment in cadrul apicatiei, nefiind eficient ca acest proces sa se execute in cadrul controllerului. Astfel, atunci cand se trimite un email, se creeaza un thread in interiorul caruia se “publica” evenimentul de trimitere a email-ului, acesta urmand sa fie trimis chiar daca aplicatia si-a continuat cursul. Expedierea este posibila importand anumite clase (prin Maven), printre care si SimpleMailMessage, care permite trimiterea de email-uri, setand inainte in proprietati datele necesare conectarii aplicatiei cu un cont de Gmail (stabilind astfel si expeditorul).





Librarii / Plugin-uri folosite

Concluzii

Lista figuri

Bibliografie

[1] Apache Cassandra Documentation 4.0, <http://cassandra.apache.org/doc/latest/>

[2] Thymeleaf – Tutorial: Using Thymeleaf, <https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html>

[3] Apache Cassandra – Wikipedia, <https://en.wikipedia.org/wiki/Apache_Cassandra>

[4] Java (limbaj de programare) – Wikipedia, <https://ro.wikipedia.org/wiki/Java_(limbaj_de_programare)>

[5] Java 8 programming for beginners: Go from zero to hero, <https://www.infoworld.com/article/3130466/java-8-programming-for-beginners-go-from-zero-to-hero.html>

[6] Spring Framework – Wikipedia, <https://en.wikipedia.org/wiki/Spring_Framework>

[7] Spring Boot Refference, <https://docs.spring.io/spring-boot/docs/2.1.5.RELEASE/reference/pdf/spring-boot-reference.pdf>

[8] Reflexion (computer programming) – Wikipedia, <https://en.wikipedia.org/wiki/Reflection_(computer_programming)>

[9] Programarea orientata pe aspecte – Wikipedia, <https://ro.wikipedia.org/wiki/Programarea_orientat%C4%83_pe_aspecte>

[10] Model-view-controller – Wikipedia, <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

[11] Girish Margan, Model-View-Controller (MVC), May 17 2018 - <https://medium.com/datadriveninvestor/model-view-controller-mvc-75bcb0103d66>

[12] Apache Tomcat – Wikipeida, <https://en.wikipedia.org/wiki/Apache_Tomcat>

[13] Maven, Apache Maven Project – Introduction to the POM, <https://maven.apache.org/guides/introduction/introduction-to-the-pom.html>

[14] Spring Framework – Introduction to the Spring Framework, <https://docs.spring.io/spring/docs/4.3.x/spring-framework-reference/html/overview.html>

[15] Distributed database – Wikipedia, <https://en.wikipedia.org/wiki/Distributed_database>

[16] Wide column store – Wikipedia, <https://en.wikipedia.org/wiki/Distributed_database>

[17] Tutorialspoint – Cassandra Tutorial, <https://www.tutorialspoint.com/cassandra>

[18] Datastax Documentation – CQL Refference, <https://docs.datastax.com/en/cql/3.3/cql/cql_reference/cqlCommandsTOC.html>

[19] Javapoint – Spring Security Tutorial, <https://www.javatpoint.com/spring-security-tutorial>

[20] Cross-site request forgery – Wikipedia, <https://en.wikipedia.org/wiki/Cross-site_request_forgery>

[21] Thymeleaf – Wikipedia, <https://en.wikipedia.org/wiki/Thymeleaf>

[22] HTML – Wikipedia, <https://en.wikipedia.org/wiki/HTML>

[23] HTML5 – Wikipedia, <https://en.wikipedia.org/wiki/HTML5>

[24] Web Techology and Trend: HTML5, <http://www-scf.usc.edu/~chenemil/itp104/webtech.html>

[25] Cascading Style Sheet – Wikipedia, <https://en.wikipedia.org/wiki/Cascading_Style_Sheets>

[26] Boostrap – Wikipedia, <https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)>

[27] JavaScript – Wikipedia, <https://en.wikipedia.org/wiki/JavaScript>

[28] Jquery – Wikipedia, <https://en.wikipedia.org/wiki/JQuery>