Universitatea Tehnică "Gheorghe Asachi" din Iași
Facultatea de Automatică și Calculatoare
Domeniul Calculatoare și Tehnologia Informației
Specializarea Tehnologia Informației

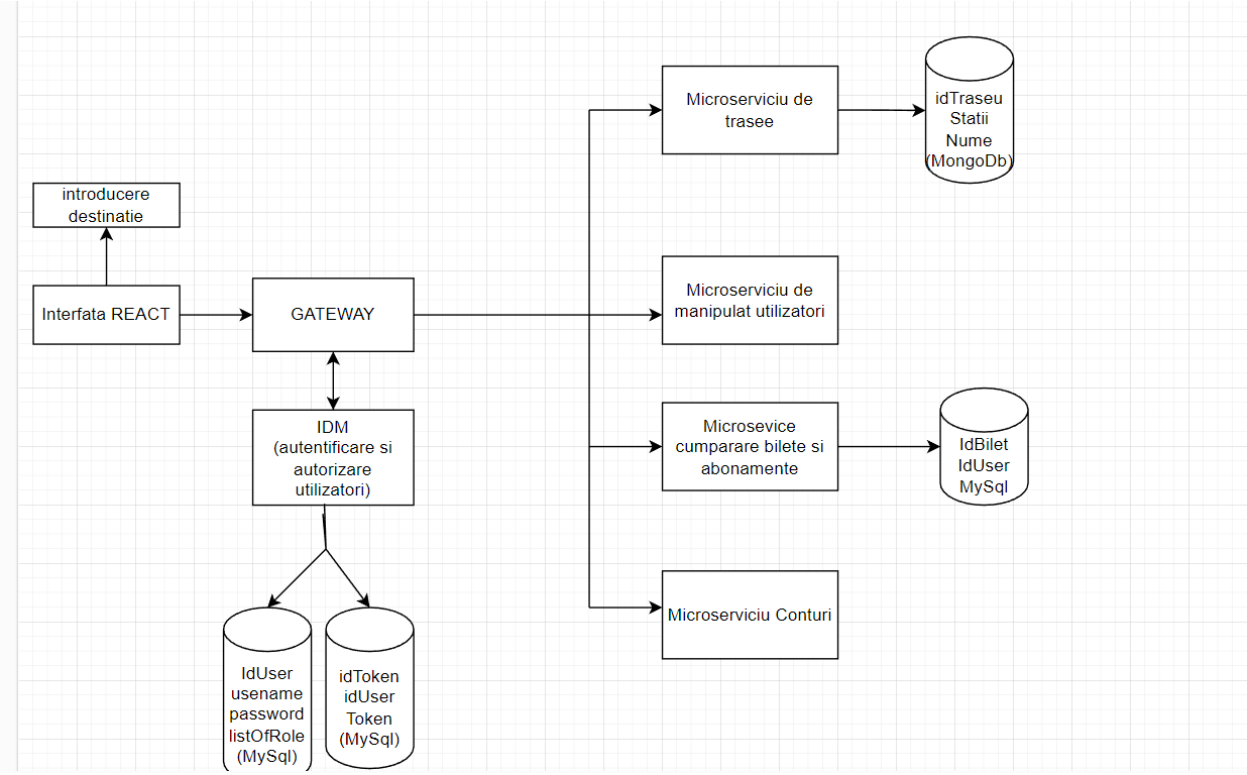
City Moving - aplicație pentru călătorit și gestionat transportul în comun

Raport intermediar

Student: Enache Ștefan
Coordonator: șef lucrări dr.ing. Cristian Aflori

1.1 Arhitectura software:

Arhitectura pe care am ales-o este o arhitectura bazată pe microservicii REST deoarece se poate dezvolta o aplicație scalabilă, cu o mentenanță facilă. Arhitectura pe microservicii REST implică dezvoltarea și implementarea unei aplicații că un set de servicii mici, independente, care comunică între ele prin intermediul interfețelor de programare a aplicațiilor (API) bazate pe protocolul HTTP.



Arhitectura pe microservicii la care m-am gândit momentan, include serviciile de utilizatori, de cumpărare de bilete și gateway, la care urmează să mai adaug încă unul de trasee și autobuze, și unul de portofel. Fiecare dintre aceste servicii este dezvoltat și implementat că un serviciu REST independent, care să aibă propriul său depozit de date (MySQL și MongoDb, în cazul meu).

Arhitectura pe microservicii presupune că fiecare serviciu să aibă propria baza de date, unde se vor salva datele necesare pentru fiecare serviciu.

Pentru aplicația mea, fiecare serviciu v-a salva date într-o baza de date specifică. De exemplu, serviciul de utilizatori v-a salva în baza de date, informații despre utilizatori precum: nume, prenume, email, username, parolă și o listă de roluri.

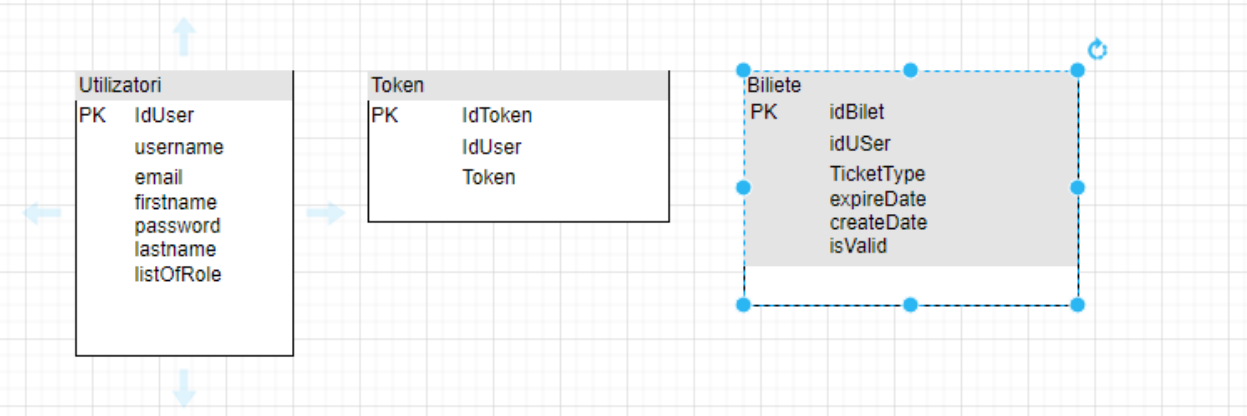
Aceste date fiind necesare în momentul autentificării în aplicație. Se vă face o căutare în baza de date, dacă utilizatorul exista, în cazul că acesta nu există, nu voi avea acces la celelalte puncte finale ale aplicației.

Servicul de gateway, v-a salva în baza de date token-ul pentru un anumit utlizator sub următoarea structură: idToken, idUser, Token.

Aceste date sunt salvate deoarece îmi sunt necesare pentru a vedea dacă un utilizator are drepturi în aplicație de a accesa un anumite puncte finale.

Servicul de cumpărat bilete, v-a salva un bilet sub următoarea formă: idBilet,idUser, TickerType, expireDate, createDate, isvalid.

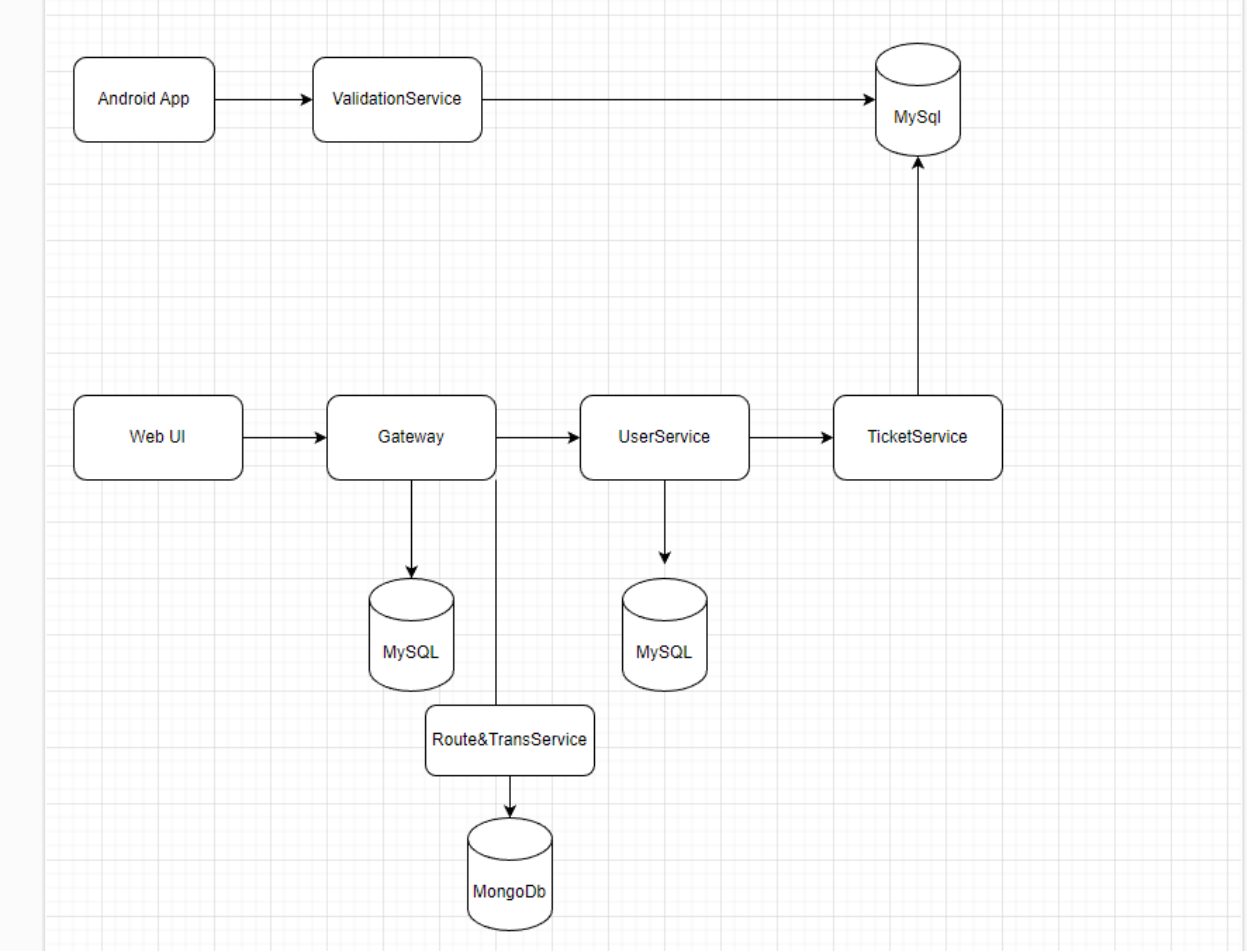
Aceste date îmi sunt necesare deoarece un utilizator își vă putea vedea ultimele 5 tranzacțiile și pentru ca utilizatorul cu rol de controlor, să aibă accesul necesar pentru a determina dacă un bilet este valid.



Interfața de utilizator este dezvoltată utilizând React, iar comunicarea dintre interfața de utilizator și poartă se face prin intermediul API-urilor REST furnizate de poartă.

Pentru a asigura securitatea serviciilor, utilizez Spring Security, care este o bibliotecă de securitate populară pentru aplicațiile Java. Aceasta o utilizez pentru a gestiona autentificarea și

autorizarea utilizatorilor care accesează serviciile menționate.



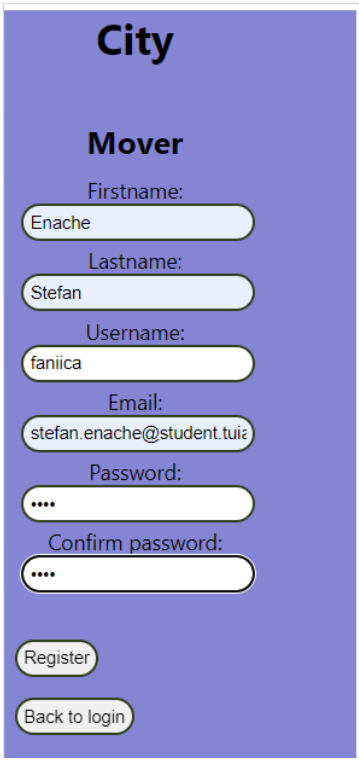
1.2 Rezultate obtinute:

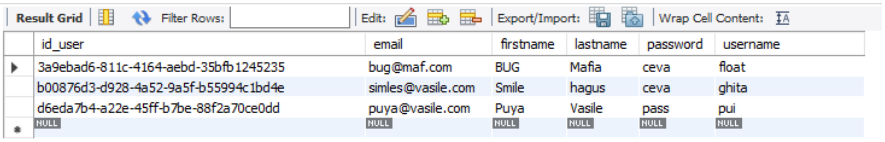
Serviciul de utilizator - Acest serviciu este utilizat pentru a gestiona informațiile despre utilizatori, cum ar fi numele, adresă de e-mail și parolă.

Serviciul de cumpărare de bilete - Acest serviciu il utilizat pentru a gestiona informațiile despre bilete și tranzacțiile de cumpărare. Acesta ar putea fi dezvoltat ca un serviciu independent, care să aibă propriul depozit de date MySQL.

Gateway - o utilizez pentru a gestiona traficul către și de la serviciile din spate. Acesta este dezvoltat că un serviciu independent, care să furnizeze o interfață unificată pentru utilizatorii finali. Gateway este utilizată pentru a proteja serviciile din spate, precum și pentru a efectua autentificarea și autorizarea utilizatorilor.

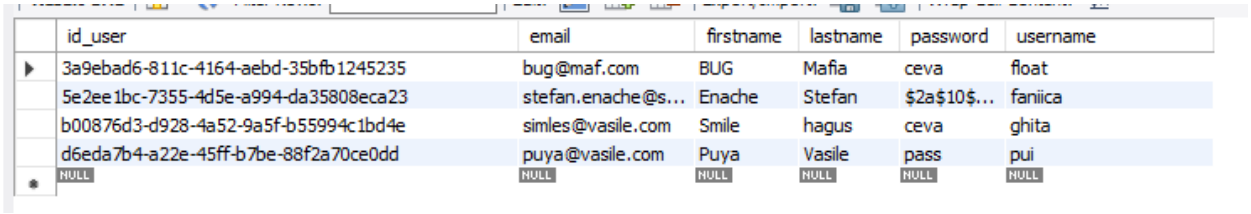
Înregistrare utilizator:





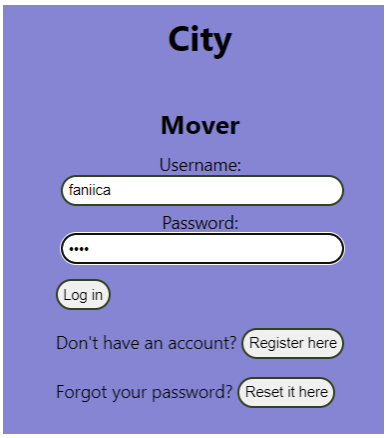
id_user	email	firstname	lastname	password	username
3a9ebad6-811c-4164-aebd-35bfb1245235	bug@maf.com	BUG	Mafia	ceva	float
b00876d3-d928-4a52-9a5f-b55994c1bd4e	simles@vasile.com	Smile	hagus	ceva	ghita
d6eda7b4-a22e-45ff-b7be-88f2a70ce0dd	puya@vasile.com	Puya	Vasile	pass	pui
NULL	NULL	NULL	NULL	NULL	NULL

Rezultatul:



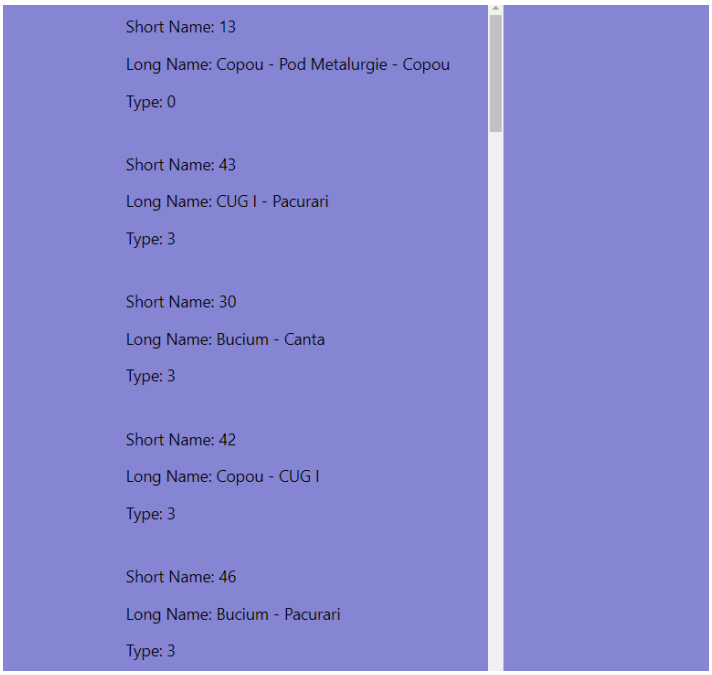
id_user	email	firstname	lastname	password	username
3a9ebad6-811c-4164-aebd-35bfb1245235	bug@maf.com	BUG	Mafia	ceva	float
5e2ee1bc-7355-4d5e-a994-da35808eca23	stefan.enache@s...	Enache	Stefan	$2a$10$...	faniica
b00876d3-d928-4a52-9a5f-b55994c1bd4e	simles@vasile.com	Smile	hagus	ceva	ghita
d6eda7b4-a22e-45ff-b7be-88f2a70ce0dd	puya@vasile.com	Puya	Vasile	pass	pui
NULL	NULL	NULL	NULL	NULL	NULL

Conectare la cont:

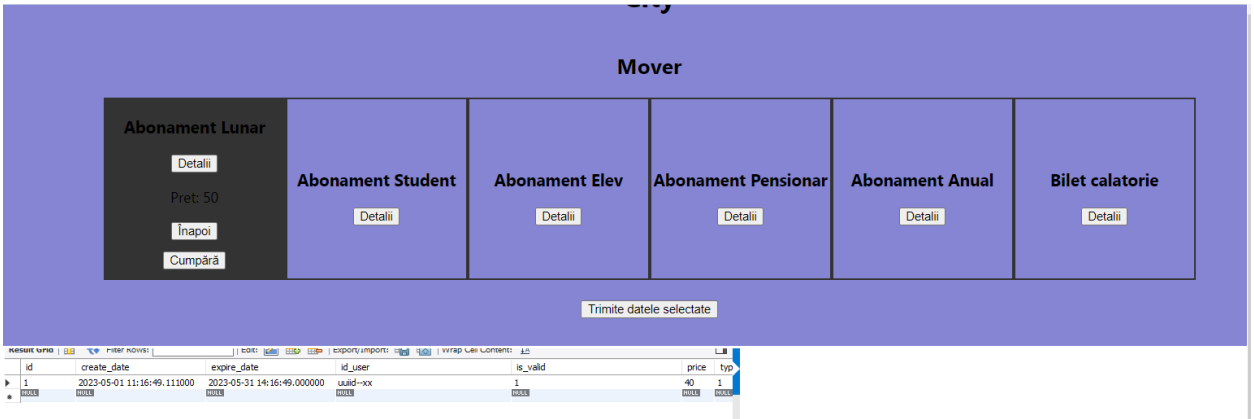


Rezultatul:

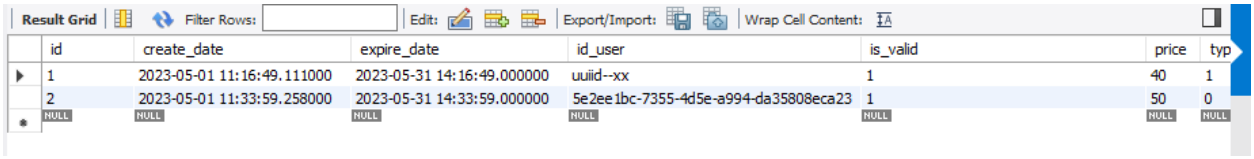
Informații despre rutele existente:



Utilizarea serviciului de cumpărat bilete:



Rezultatul:



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	create_date	expire_date	id_user	is_valid	price	typ
1	2023-05-01 11:16:49.111000	2023-05-31 14:16:49.000000	uuiid--xx	1	40	1
2	2023-05-01 11:33:59.258000	2023-05-31 14:33:59.000000	5e2ee1bc-7355-4d5e-a994-da35808eca23	1	50	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

1.3 Dificulati in implementare:

Complexitatea - Implementarea acestei arhitecturi pe microservicii poate fi destul complexă, deoarece serviciile sunt separate și comunică între ele prin intermediul interfețelor API. Acest lucru poate duce la o mai mare complexitate în dezvoltare, testare și implementare. De exemplu, o problemă potențială este asigurarea coerenței datelor între servicii. Deoarece fiecare serviciu are propriul său depozit de date, asigurarea coerenței între datele stocate în fiecare serviciu este o provocare.

Autentificare și autorizare - Implementarea unei autentificări și autorizări adecvate în arhitectură pe microservicii este suficient dificilă. Deoarece fiecare serviciu trebuie să fie securizat individual, este important să am o strategie bine definită pentru a gestionarea autentificarea și autorizarea în toate serviciile. În plus, este important să se asigure că datele tranzacționate între servicii sunt securizate și criptate.

Managementul tranzacțiilor - În arhitectura pe microservicii, tranzacțiile trebuie să fie gestionate la nivelul serviciilor individuale. Aceasta au fost o provocare, deoarece fiecare serviciu a fost proiectat pentru a avea propriul său mecanism de gestionare a tranzacțiilor.

Comunicarea între servicii - Comunicarea între servicii este realizată prin intermediul interfețelor API. În cadrul aplicației am interfețe API bine definite și actualizate pentru a asigura comunicarea eficientă între servicii. De asemenea, m-am asigurat că interfețele API sunt compatibile între servicii și că datele sunt tranzacționate într-un format standardizat.

Gestionarea erorilor - Gestionarea erorilor este destul de dificilă într-o arhitectură pe microservicii decât în monolit. Este destul de important să am un sistem bine definit pentru gestionarea erorilor și a monitorizării pentru a asigura disponibilitatea serviciilor. În plus, trebuie să existe un sistem de înregistrare a evenimentelor pentru a ajuta la diagnosticarea și remedierea erorilor.

Dificultăți în implementare am întâlnit la securizarea aplicației și mai exact la distribuirea și autorizarea rolurilor pentru punctele de intrare în aplicație. De asemenea, alt punct dificil a fost integrarea și conectarea părții de frontend cu backend-ul. O altă dificultatea a fost integrarea Spring Sercurity pentru generearea de token atât la autentificare cât și la autorizare.