Bontida part 3

Student: Gabor George Cătălin
Grupa: 30239

May 1, 2018

Cuprins

1 Analiza cerințelor

1.1 Specificare cerinte

Se cere realizarea părții de "Back-end" pentru o aplicație mobile pentru a oferi informații participanților la fectivalul Electric Castle.

Aplicația trebuie să fie construită pe baza arhitecturii Service Oriented Architecture (SOA), avûand un serviciu Web principal care expune un API aplicației mobile și două servicii ce vor fi folosite de cel principal. Primul va returna informații legate de artiști iar al doilea informații legate de vreme.

1.2 Cerinte functionale

Aplicația trebuie să pună la dispoziție utilizatorilor următoarele funcționalități:

- GET http://localhost/api/artists
  - Returneaza o lista cu toate trupele
- GET http://localhost/api/artists?stage=<stage_name>
  - Returneaza o lista cu toate trupele care vor canta la scena <stage_name>
- GET http://localhost/api/artists?day=<day_nr>&hour=<hour(5-10)>
  - Returneaza o lista cu toate trupele care vor canta la data si ora respectiva
- GET http://localhost/api/artists?artistname=<artist_name>
  - Returneaza scena, ziua, ora + vremea de la ora respectiva

1.3 Cerinte non-functionale

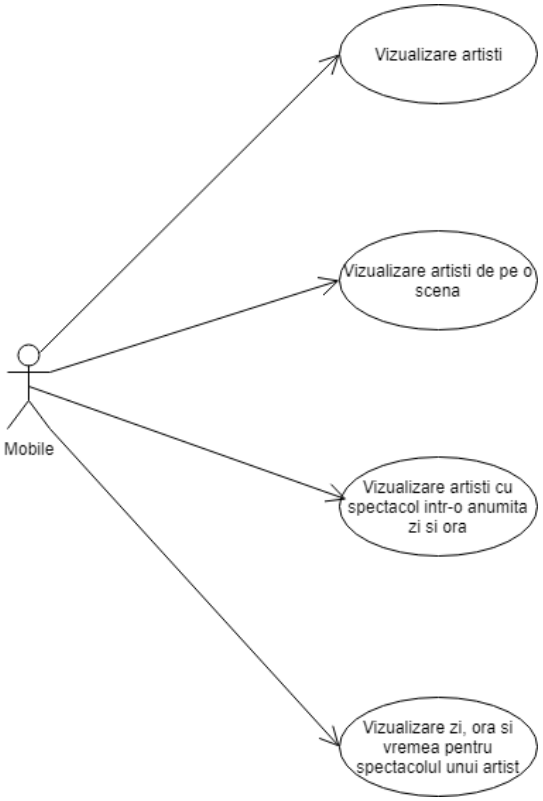
Urmatoarele cerinte trebuie respectate:

- utilizarea unui framework ce implementează patternul MVC (Spring Boot);
- serviciul principal trebuie implementat prin agregarea a două servicii secundare;
- servicile secundare vor fi create utilizând https://sheetsu.com/;
- utilizarea pattern-ului Chain of Responsibility;
- utilizarea arhitecturii SOA;
- implementarea funcționalității de cache cu timp de expirare a datelor asupra informaților returnate de servicile secundare;
- utilizare https://sheetsu.com pentru crearea API de returnare date despre artiști și vreme.

2 Use Case Model

Use case: Vizualizarea artiștilor ce au spectacol pe o scenă;
Level: User-goal;
Actor princital: Dispozitivul mobil;
Scenariu de succes: În browser este introdusă adresa http://localhost:8080/api/artists?stage=<User_Input >. Aplicația va căuta în cache informațile și dacă nu sunt prezente sau a trecut suficient timp va face o cerere la serviciul ce returnează informații despre artiși. Apoi se face o filtrare a artiștilor dupa scenă și se returnează numelor celor ce corespund.
Extensii: Operatia poate eșua dacă:

- intervine o problemă cu serviciul de informații pentru artiști.



3 Design-ul arhitectural al sistemului

3.1 Descrierea modelului arhitectural

Este utilizat pattern-ul arhitectural MVC.
Aplicatia este compusa din două nivele:

- Controller - leaga Modelul de View și procesează toate cererile;

- Model - oferă acces structurat la date.
  - Entities - structurile de date;
  - Services - combină Entities și Repositories pentru a returna date structurate.

Deasemenea este utilizat Service Oriented Architecture, aplicația utilizând două servicii:

- EC_artists - access informații artiști;
- EC_weather - access infromații vreme.

3.2 Diagrame

Urmatoarea diagrama prezinta structura pachetelor din sistem :

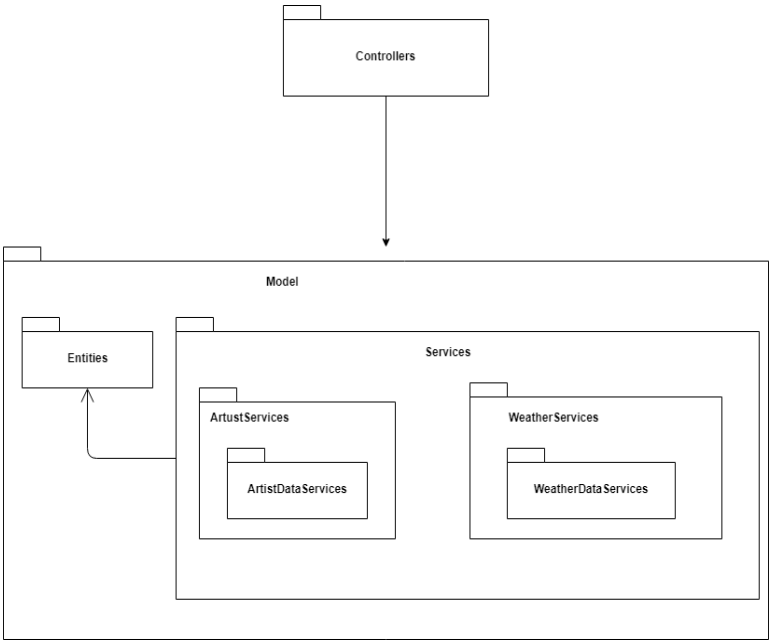


Diagrama de componente:

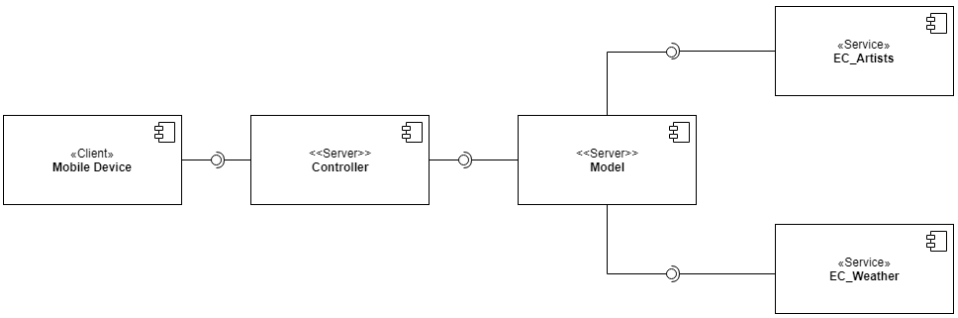
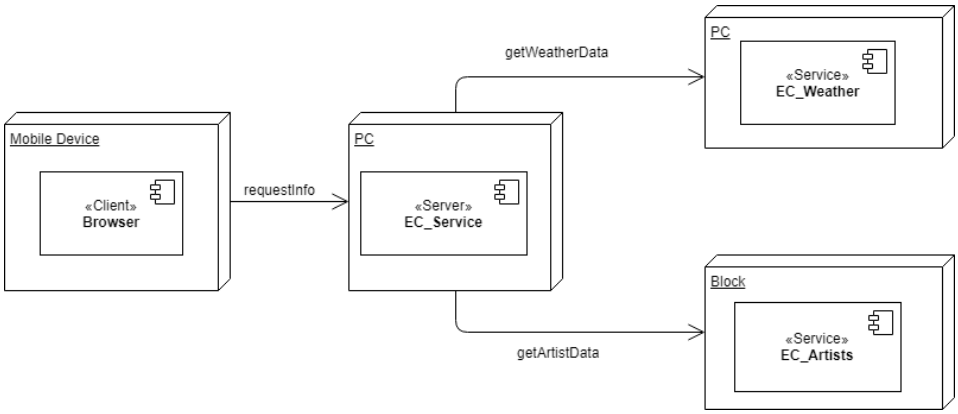
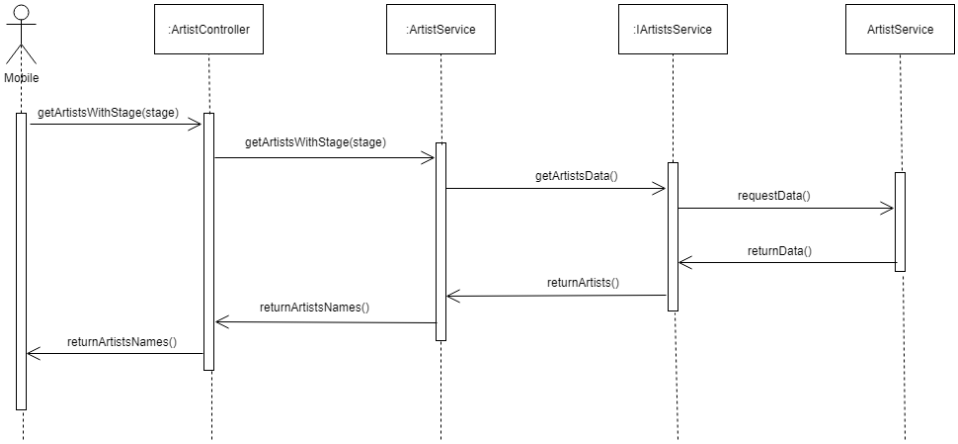


Diagrama de deployment:



4 Diagrame de secventa UML

Consideram scenariul în care se face o cerere de returnare a artiștilor ce au spectacol pe o anumită scenă:



5 Design-ul claselor

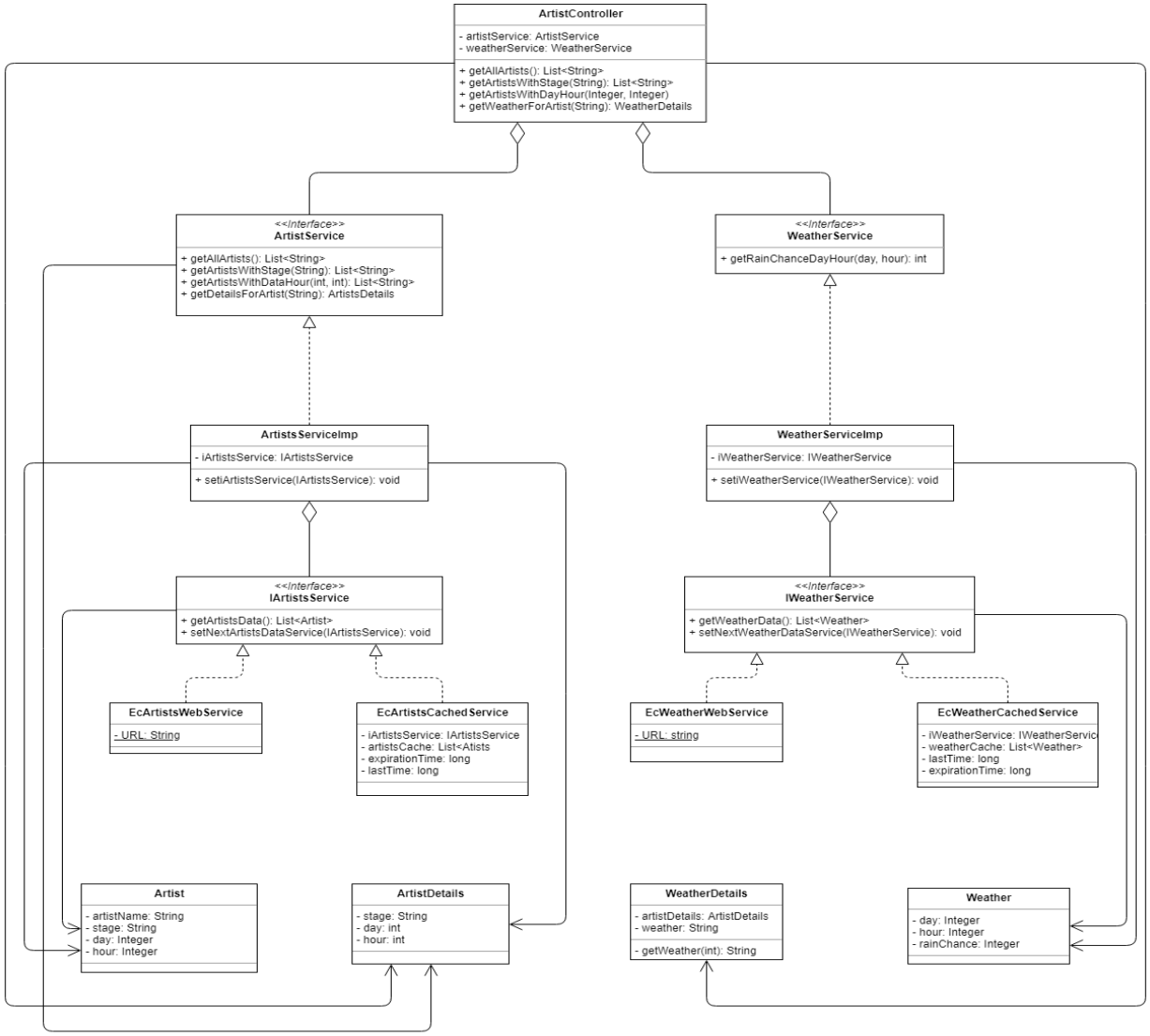
5.1 Descriere design pattern-uri utilizate

Au fost utilizate urmatoarele:

- Chain of responsability - Clasa EcArtistsCachedService returnează în mod normal datele despre artiști, dau uneori cere clasei EcArtistsWebService datele pentru a le reîmprospăta. Comunicarea se face utilizând interfața IArtistsService.

- Composite - WeatherDetails are ca atribut o instanță a clasei ArtistDetails;

5.2 Diagrama de clase



6 Model de date

Datele au fost stocate utilizând Google Spreadsheet. Tabelele având următoarea formă:

ArtistName	Stage	Day	Hour
Jessie J	Main	1	5
Cancer Bats	Main	1	8
Jp Cooper	Main	1	10
Raresh	Main	2	5
Nothing But Thie	Main	2	10
Subcarpati	Main	3	9
London Gramma	Main	3	10
Damian Marley	Red Bull	1	5
Excision	Red Bull	1	6
San Holo	Red Bull	2	6
Wolf Alice	Red Bull	2	7
Elrow	Red Bull	2	10
Vita de Vie	Red Bull	3	10
Mura Masa	BT	1	5
Richie Hawtin	BT	1	6
High Contrast	BT	2	6
Son Lux	BT	2	7
Mum	BT	2	9
Alison Wonderlar	BT	2	10

Day	Hour	RainChance
1	5	49
1	6	2
1	7	16
1	8	30
1	9	93
1	10	46
2	5	41
2	6	53
2	7	41
2	8	63
2	9	70
2	10	52
3	5	86
3	6	66
3	7	83
3	8	2
3	9	52
3	10	60

7 Testarea sistemului

S-a folosit o strategie de testare incrementală, funcționalitățile au fost verificate pe măsură ce au fost adăugate in sistem. Astfel, principalele metode de testare au fost:

- Unit testing - au fost create două pachete TestControlServices(verificare prelucrarea datelor) și TestGetDataServices(verificare returnare date de la servicii);
- data-flow (au fost date anumite intrări si s-a observat raspunsul sistemului);
- white-box (codul a fost disponibil).

8 Bibliografie

Pentru utilizare Spring Boot și Servicii Rest:

- `https://spring.io/guides/gs/rest-service/`
- `https://spring.io/guides/gs/consuming-rest/`

Pentru anumite erori:

- `https://stackoverflow.com/`