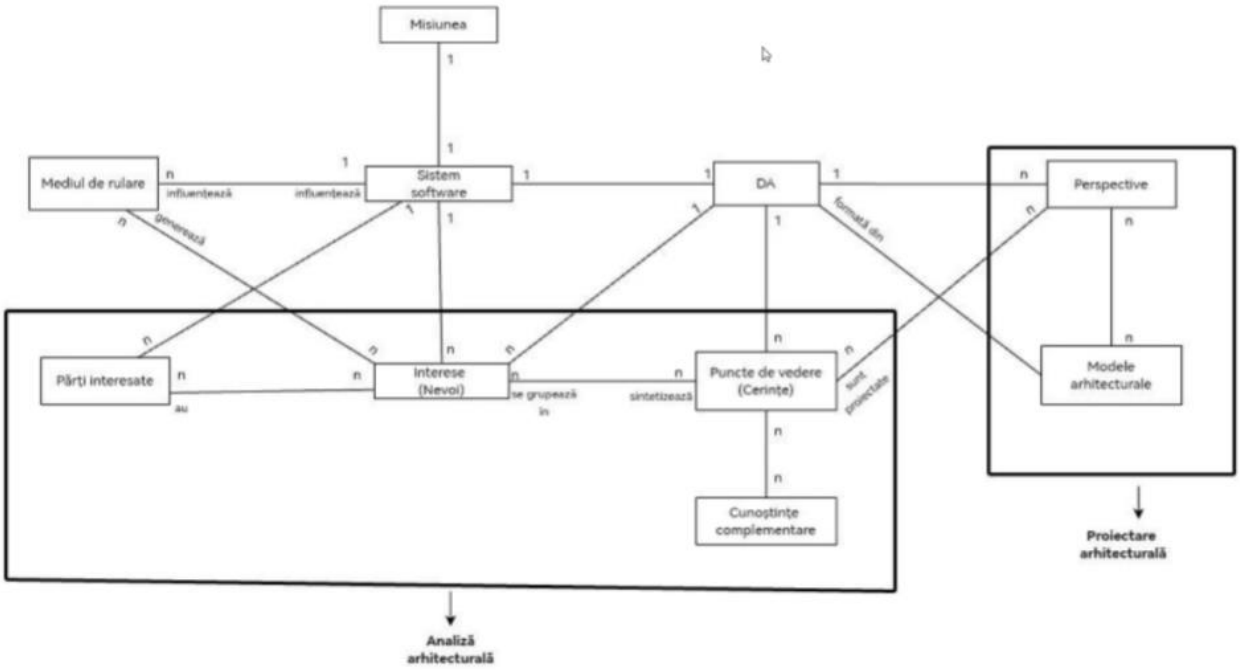
# 1. Curs 1

## Def:

Arhitectura unui system software reprezinta componentele **principale** ale sistemului, relatiile dintre ele, relatiilor acestora cu mediul in care ruleaza sistemul, precum si strategia de dezvoltare si intretinere a sistemului pe totalitatea ciclului lui de viata.

Arhitectura unui sistem software reprezinta suma deciziilor **majore** pe care le luam in legatura cu proiectarea, implementarea si intretinerea sistemului pe tot ciclul lui de viata.




Scopul arhitecturii este de a defini:

- relațiile dintre componentele principale ale sistemului (care depind de complexitatea aplicației)
- relațiile sistemului cu mediul în care va fi utilizat
- modul în care ajungem la implementarea aplicației pe baza arhitecturii propuse (fezabilitatea arhitecturii

Proiectarea:

1. arhitecturală ( la nivel centralizat) - arhitect software
2. componentelor (la nivel de programator)

Analiza:

1. arhitecturală (la nivel centralizat) - analist tehnic
2. componentelor (la nivel de analist / programator)
3. funcțională (de business)
4. tehnică (calitățile aplicației)

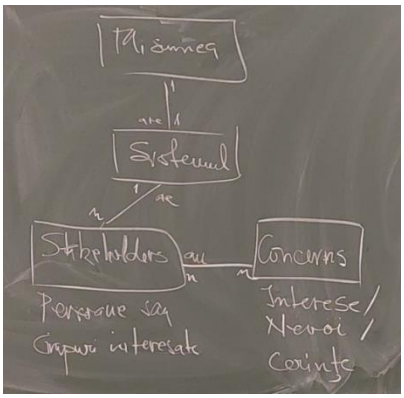
# 2. Curs 2 – Introducere + Tipuri de cerinte

Arhitectura are 2 parti

- Analiza arhitecturala – 'Ce nevoi trebuie sa satisfaca aplicatia dpdv arhitectural'
- Proiectarea arhitecturala – 'Cum acopera aplicatia nevoile sale arhitecturale'

Stakeholders:

- Utilizatori
- Echipe de dezvoltare
- Clienti + sponsori
- Concurenta




**Ex** Tipuri de utilizatori:

- Donator
- Promotor
- Admin
- System

| | CINE? | CE? | DE CE? |
|---|---|---|---|
| **US1** | Donatorul | Susține unele cauze sociale | **Misiunea aplicației** |
| **US2** | Promotorul | Promovează cauze sociale | |
| **US3** | Admin | Asigură climatul de încredere în aplicație | |
| **US4** | Sistemul | Furnizează în baza datelor caracteristici avansate | |

## Misiune – obiectivele principale ale utilizatorilor

Ex **misiune**: Cresterea numarului de cause speciale sustinute si sumele donate pentru aceste cauze.

## Ex Nevoi / probleme

- Nu exista o cultura a donatiei in societatea romaneasca.
- Exista discrepanta intre cauzele sustinute de organizatii sau personalitati influente
- In general, donatorii nu sunt informati de cum le sunt utilizate fondurile contribuite
- Exista riscul unor inselaciuni, sau cauze inventate

## Colectarea cerințelor (intereselor, nevoilor)

Principiul 1: Nicio aplicație software nu reprezintă un scop în sine. Orice aplicație software reprezintă soluția la o anumită problemă din lumea reală.
Principiul 2: Cerințele aplicației software nu se inventează (nu presupunem că ar fi într-un anumit fel). Cerințele se extrag de la persoanele sau grupurile interesate (Stakeholders)
Principiul 3: Cerințele colectate de la părțile interesate se curăță, se completează cu informații specifice colectate din literatura de specialitate, aplicații similare și se structurează în Puncte de Vedere (Viewprints)
Principiul 4: Nu toate cerințele au aceeași relevanță pentru dezvoltarea aplicației. Cerințele trebuie prioritizate

Cerintele sunt:

- *Functionale*

- Sistemul trebuie sa ofere posibilitatea utilizatorilor de a face rezervari la restaurantele din aplicație:
  - Clienții apreciază opțiunea de a putea efectua rezervări din același loc de unde se informează [1]
  - Clientul se bucură de flexibilitate, putând efectua rezervări în afara programului de lucru al restaurantelor, fapt văzut ca un avantaj de către acest grup [1]
  - Mesele dorite vor fi selectate de pe harta restaurantului, fapt ce oferă flexibilitate [2]

- *Non-functionale*:

  - X **de calitate** (legate de: Utilizabilitate, Testabilitate, Siguranta, Performanta, Disponibilitate si modificabilitate) **Ex**: Google este rapid si precis (performance) cu interfata simpla (usability)

,

Sistemul trebuie sa fie cat mai ușor de folosit, oferind flow-uri intuitive, care minimizează fricțiunea dintre client și aplicație, pentru a ușura procesul de selectare a restaurantului dorit, deci a îndeplinirii misiunii aplicației. (Utilizabilitate)

Un specialist în testare, un stakeholder, sau chiar un dezvoltator ar trebui să poată verifica corectitudinea output-ului, consistența în răspunsuri, performanța, securitatea și alte atribute funcționale sau non-funcționale ale sistemului. [12] (Testabilitate)

  - X **constrangeri** – Ex: lipsa de buget, echipe de dimensiuni mici, lipsa de experienta a membrilor, lipsa de timp

# 3. Curs 3 – Obiective, user stories, quality scenarios si use cases

cerintele functionale au use case uri ca sa fie explicate in detaliu
cerintele de calitate au scenarii de calitate ca sa fie explicate in detaliu

Obiective principale**:** fiecare rol genereaza un obiectiv (sau mai multe)

Ex obiective/cerinte high-level:

Ob1: Donatorul acceseaza cauzele sociale si face donatii **pentru realizarea misiunii**
Ob2: Promotorul sustine atragerea de fonduri pentru cauzele sociale **pentru realizarea misiunii**
Ob3 Adminul asigura climatul de incredere in utilizarea platformei **pentru realizarea misiunii**
Ob4 Sistemul ofera utilizatorilor functionalitati avansate **pentru realizarea misiunii**

User stories (US) o modalitate de definire a specificatiilor in domeniul software

3 componente – Entitate (cine), Actiune (ce) si scop (de ce)

Pot fi structurate in tabel sau narativ.

EX: **US1** - In calitate de client (autentificat sau nu) doresc sa vad o lista cu toate restaurantele din system pe pagina principala, pentru a imi fi facilitat procesul de alegere a unui restaurant din zona mea.

Scenariile de calitate (QS) – elemente de specificare a cerintelor de calitate - : Utilizabilitate, Testabilitate, Siguranta, Performanta, Disponibilitate si Modificabilitate

Artefactul implicat – Sistemul sau o anumita componenta (Ex: interfata)

Starea artefactului – neutra sau specifica (Ex: dupa intalnirea unei erori)

Evenimentul care declanseaza raspunsul artefactului

Strategia de raspuns - Raspunsul artefactului

Masura raspunsului – modul de masurare si valorea tinta a indicatorului

Performanță

| | |
|---|---|
| Codificare | QS3 |
| Denumire | Performanță |
| Inițiator | Utilizator |
| Declanșator | Userul accesează aplicația și realizează operații |
| Artefact | Sistem |
| Stare | Aplicația rulează pe un mediu de producție sau un mediu asemănător |
| Răspuns | În urma operațiilor se obține un răspuns în timp util |

| | |
|---|---|
| Măsură/Indicator | Pentru retenția utilizatorilor ar trebui ca timpul de răspuns pentru orice operație să se simtă natural, adică să nu pară că se încarcă mai mult decât ar trebui. |

Atentie!!!!! Masura din exemplu e gresita. Un exemplu mai bun:

Masurat prin timpul de raspuns al oricarei actiuni. Valoarea tinta este de maxim 3 secunde/actiune.

Cazuri de utilizare (UC) – o secventa de pasi in aplicatie pe care **un anumit** tip de utilizator le executa pentru a-si indeplini **un anumit scop** (reflectat de un obiectiv al utilizatorului in aplicatie)

| | |
|---|---|
| Codificare | UC1 |
| Denumire | Sistemul oferă utilizatorului posibilitatea de a vizualiza restaurantele înscrise în aplicație |
| Rol principal | Utilzator |
| Roluri secundare | - |
| Precondiții | Utilizatorul a accesat aplicația (nu este nevoie de autentificare) |
| Declanșator | Utilizatorul dorește sa aleagă un restaurant |
| Scenariu de succes | 1. Utilizatorului i se afișează lista completa cu restaurante<br>2. Selectarea unui restaurant permite vizualizarea corectă a detaliilor |
| Postcondiții | Informațiile sunt corecte și vizibile în aplicație |
| Șcenariu de eșec | - |
| Extensii | 1. Adăugarea de filtre inteligente pe lista de restaurante<br>2. Sistem de recomandare în funcție de preferințele utilizatorului |

# 4. Curs 4 – Cerinte de calitate – Performanta

Clase de cerinte de calitate:

1. Performanta
2. Disponibilitate
3. Securitate
4. Utilizabilitate/UX
5. Modificabilitate
6. Scalabilitate (cresterea in volum sau adaugarea de functii noi)
7. Portabilitate (capacitatea de a rula la nivelul asteptat de calitate in dif. medii)
8. Testabilitate
9. Mentenabilitatea (actualizari, costuri de operare, etc)
10. Accesibilitate (subcateg. UX)

## Calitatile unui sistem – Caracteristici care ne diferentiaza sistemul in raport cu alte sisteme similar

### Performanta

timp mic de raspuns, conservarea resurselor consumate (perf. Economica), relevanta raspunsurilor

Artefacte: system, sau o anumita componenta – Ex Serviciu de recomandari

Stare: - (context). nr de utilizatori concurenti mare, volum de date procesate, etc

Eveniment declansator: solicitarea arterfactului

Strategii: o anumita clasa de algoritmi

Masura raspunsului: indicator specific clasei de algoritmi SAU satisfactia utilizatorilor

# 5. Curs 5 – Cerinte de calitate – Disponibilitate

La revenirea dintr-o stare de nefunctionare se iau in considerare doar strategiile implementate in arhitectura aplicatiei, nu si interventiile exterioare (ex: Have you tried turning it off and on again)

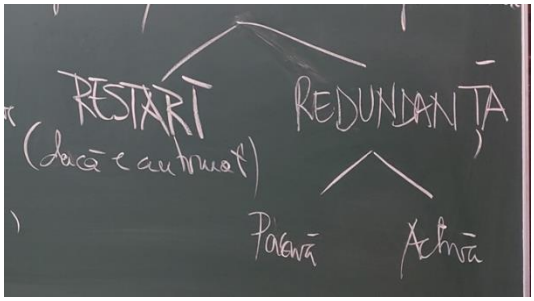
Artefactul: sistemul

Starea: normala sau afectata

Evenimentul declansator: solicitare adresata artefactului

Strategie de raspuns: Detectia/notificarea/revenirea din indisponibilitate

Masuri: procent solicitari raspunse cu success, timp de functionare neintrerupta/durata intreruperi



# 6. Curs 6 – Cerinte de calitate – Securitate si Utilizabilitate

## Securitate

Autentificare si autorizare; Nivelul de Securitate se decide in functie de riscurile existente (ex: nu iti trebuie protectie pentru SQL Injection intr-o aplicatie fara DB SQL)

Artefactul: sistemul

Starea: normala sau afectata

Evenimentul declansator: atac asupra artefactului

Strategie de raspuns: Preventia/Detectia/Revenirea din atac

Masuri: procent preventive/ timp de detective/ timp de revenire

## Utilizabilitate

Usurinta de utilizare; depinde mult de profilul utilizatorilor; usurinta de utilizare trebuie sa fie proportionala cu disponibilitatea de invatare a utilizatorilor (Ex: daca faci o aplicatie de unde isi ia bunica retelele nu vrei sa contina mai mult de 2 butoane)

Artefactul: Interfata

Starea: profilul utilizatorilor + **?Legacy?** (Ex: un utilizator bleg)

Evenimentul declansator: Un anumit context de utilizare (Ex: Utilizarea meniului principal pentruj a adauga o comanda)

Strategie de raspuns: Simplitate/intuitivitate

Masuri: numarul de pasi/timpul de finalizare a taskului/numarul de finalizari esuate

# 7. Curs 7 – Descompunerea graduala a specificatiilor

Strategii din seminar:

I Confirmarea donatiilor dupa efectuare (proc. de util nemultumiti < 1%)

II Distribuirea automata a informatiilor catre donatori cand promotorul actualizeaza campania (cresterea numarului de donatii la 6/an sau cu 20%)

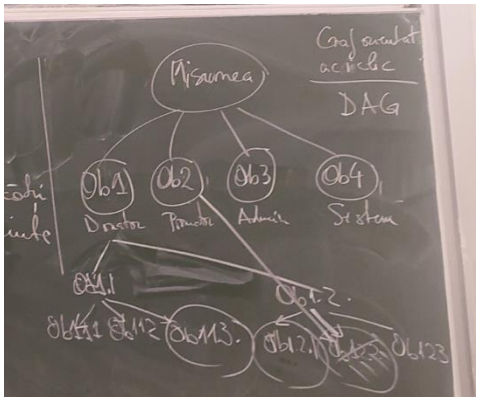
III Notificarea promotorilor cand a trecut o anumita perioada de timp de la ultima actualizare (legata de II, vrem ca 80% din promotor isa actualizeze informatiile)

Misiune:

Cresterea numarului de cause speciale sustinute si sumele donate pentru aceste cauze.

### Descompunerea graduala a specificatiilor

- Vrem sa ajungem de la misiune la cerintele de calitate principale.
- Specificatiile sunt scrise sub forma de user stories.
- Descompunem un US transformand componenta centrala in scopul subaspectelor ei




Obiective:

Ob1: Donatorul **doneaza catre cauzele alese** pentru realizarea misiunii

Ob1.1 Donatorul **doneaza o data pentru o cauza** pentru a **dona catre cauzele alese**

Ob1.1.1 Donatorul allege o cauza ca sa doneze pentru ea, in mod satisfacator

Ob1.1.2 Donatorul face o plata online ca sa doneze pentru o cauza, in mod satisfacator

Ob1.1.3 Sistemul confirma plata catre donator pentru **ca acesta sa doneze in mod satisfacator**-> Scenariu de calitate I

Ob1.2 Donatorul **doneaza repetat pentru o cauza aleasa** pentru **a dona catre cauzele alese**

Ob1.2.1 Sistemul notifica periodic promotorul despre actualizarea informatiilor pentru ca donatorul sa doneze repetat pentru cauza => scenariul de calitate III

Ob1.2.2 Promotorul actualizeaza periodic informatiile despre cauza pentru ca donatorul sa doneze repetat pentru ea => scenariul de calitate II

Ob1.2.3 Donatorul acceseaza informatiile actualizate pentru **a dona repetat**

Ob2: Promotorul sustine atragerea de fonduri pentru cauzele sociale **pentru realizarea misiunii**

Ob3 Adminul asigura climatul de incredere in utilizarea platformei **pentru realizarea misiunii**

Ob4 Sistemul ofera utilizatorilor functionalitati avansate **pentru realizarea misiunii**

## 8. Curs 8 – Proiectare arhitecturala - Scenariu vs tactica

Analiza scenariilor de calitate duce la definirea unor QS, QS detin strategii.

Orice strategie se implementeaza prin intermediul a una sau mai multe solutii practice care sunt definite de anumite modele de implementare, numite si **tactici arhitecturale** de raspuns.

Exemple de tactici din curs: gamification (transformarea unui flow normal in unul ce ofera gratificare, cum ar fi folosirea a 50% din donatii pentru cauze si 50% intr-un pool de tip LOTO), feedback (puternic dezvoltat mai sus), asistenta promotorilor prin diferite unelte.

## 9. Curs 9 – Perspective arhitecturale

Arhitectura se prezinta din diferinte puncta de vedere, numite si perspective. Majoritatea perspectivelor pot fi reprezentate prin UML.

- Utilizare - wireframe-uri
- Datelor - ERD-uri
- Structurala - Class diagram
- Comportamentala - Activity sau interaction diagram

- Fizica - la nivel de componente fizice – necesita x servere, PC strasnic etc.

Optional

- Securitate - autentificare si autorizare
- Gestionare a resurselor
- Dezvoltare
- Testare

Majoritatea cerintelor de calitate pot fi privite din multiple perspective de calitate.
Ex:
Feedback-ul tine atat de utilizare si date cat si structura si comportament.
Gamification tine mai mult de comportament si structura dar poate fi influentat si de date si utilizare.

# 10. Curs 10 – Pasii de proiectare ai arhitecturii

1. Nicio aplicatie software nu reprezinta un scop in sine, ci este folosita ca metoda de a atinge un scop.
2. Exista posibilitatea sa nu intelegem de la inceput foarte bine ce avem de facut
3. In cazul in care estimam o alterare semnificativa a ipotezelor pe parcursul dezvoltarii atunci devin relevante calitati arhitecturale precum modificabilitatea si testabilitatea
4. Misiunea aplicatiei, odata stabilita, genereaza tot proiectul de dezvoltare si reprezinta cel mai important mod de validare a rezultatului final
5. Selectia si prioritazarea cerintelor in raport cu marimea si obiectivele principale sunt esentiale in privinta modului de proiectare, implementare si testare a aplicatiei
6. Trasabilitatea specificatiilor atat in sus catre misiune cat si in jos catre task-urile propriu zise este foarte importanta
7. La orice problema pot exista mai multe solutii iar tacticile arhitecturale pe care le alegem trebuie sa realizeze impreuna cea mai solutie la cerintele date
8. Scopul abordarii acestor perpective este acela de a ne asigura ca toate specificatiile de analiza sunt acoperite in cel putin un mod, iar toate tacticile arhitecturale adaptate sunt justificabile de cerinte si se integreaza bine intre ele
9. Se vor folosi acele perspective care include tacticile arhitecturale corespunzatoare specificatiilor de analiza evidentiate (nu neaparat tacticile enumerate la curs, dar poate altele mai relevante)
10. Alegerea tehnologiilor (a platformei / lor de dezvoltare) implica nu doar utilizarea unor biblioteci dar si a unor arhitecturi de referinta care nu reprezinta chiar arhitectura noastra, dar o 'fixeaxa in proportie de 80%' – Ex: daca faci un API in ASP.NET e recomandat sa respecte pattern-ul

# 11. Curs 11 – Stiluri arhitecturale

Fiecare tehnologie e implementata pe o anumita arhitectura (de referinta)

Arhitectura de referinta defineste o clasa de aplicatii (cam toate app dezv. In tehnologia respective)

Alegerea tehnologiei trebuie sa tina seama de specificatiile de analiza si proiectare, respective toate scenariile de utilizare, de calitate si constrangeri, precum si de stilurile arhitecturale folosite de arhitectura respectiva si calitatile arhitecturale.

## Stiluri arhitecturale:

- Client-server (clientul comunica cu serverul prin intermediul unui canal, de obicei HTTP)
- Pipes and filters (workflows, se urmeaza un chain predefinit)
- Event based (componentele dau trigger la functionalitate prin eventuri)

# 12. Curs 12 – Alegerea tehnologiilor

Tehnologia furnizeaza o mare parte (~80%) din arhitectura aplicatiei. Orice framework folosit va avea o arhitectura proprie, de baza, care desemneaza clasa aplicatiei.

Toate modificarile arhitecturale trebuie sa fie compatibile cu framework-ul

Procesul de selectie

1. Se pleaca de la descrierea arhitecturala sub forma perspectivelor relevante
2. Se considera framework-urile care acopera sau se apropie semnificativ dpdv. architectural de descrierea facuta din fiecare perspectiva
    a. Se porneste de la perspectiva fizica
    b. Se selecteaza mai multe framework-uri pe post de candidati; daca nu exista candidate, se ajusteaza descrierea arhitecturala
    c. Se compara candidatii in functie de constrangeri: - experienta echipei de dezvoltare, bugetul de infrastructura, etc.

# 13. Exemple din proiecte – Analiza arhitecturala

**1. Scopul aplicației**

Descrierea tipului, misiunii și utilizatorilor aplicației (max. ½ de pagină)

"Restaurant booking platform" este o aplicație creată pentru efectuarea rezervărilor la restaurante. Aceasta conectează de la distanță utilizatorii cu personalul restaurantelor, simplificând astfel procesul de rezervare al meselor, prin eliminarea necesității de a apela sau vizita locația dorită. De asemenea, proprietarii restaurantelor au posibilitatea de a oferi diferite facilități utilizatorilor care și-au demonstrat loialitatea față de locațiile lor, cum ar fi oferte speciale, discounturi sau diferite programe de loialitate.

Aplicația se încadrează în categoria "booking apps" fiind o aplicație web, care poate fi accesată atât de pe desktop, cât și de pe smartphone sau tabletă, indiferent de sistemul de operare sau de tipul de browser utilizat.

Misiunea este să ușureze procesul rezervării unei mese la restaurantele preferate. Întregul proces devine mai accesibil pentru toate părțile implicate, deoarece sunt eliminate apelurile telefonice sau vizitele în persoană, rezervarea putând fi efectuată în orice moment al zilei, fără a fi implicate direct, în același timp, toate părțile participante.

Grupul țintă este format din persoanele care preferă să se bucure de o masă în oraș la restaurantul preferat și doresc să rezerve cât mai ușor o masă în prealabil. Aplicația este deosebit de populară în rândul persoanelor ocupate care nu au timp să sune sau să viziteze un restaurant pentru a face o rezervare. Aceasta poate fi utilizată și de către turiștii străini, care nu sunt familiarizați cu orașul sau locația și doresc să găsească un restaurant potrivit pentru nevoile și gusturile lor, prin consultarea meniului și al recenziilor

**2. Aria de acoperire a aplicației**

"Restaurant booking platform" permite utilizatorilor să navigheze printr-o listă de restaurante, să vizualizeze meniurile, să citească recenzii de la alți utilizatori, dar și să evalueze și să facă rezervări în timp

real. Utilizatorii pot selecta data, ora și masa de pe harta restaurantului, pentru care ulterior vor fi nevoiți să confirme prezența. De asemenea, platforma le oferă clienților și alte caracteristici suplimentare, cum ar fi programele de loialitate, reducerile și ofertele speciale.

Platforma nu constituie o aplicație care doar oferă informații despre restaurante, cum ar fi meniurile, locațiile și detaliile de contact ale acestora, cât aceasta a fost dezvoltată să permită în primul rând utilizatorilor să rezerve mese la restaurantul dorit, într-o anumită dată și la o anumită oră. În mod similar, nu este o aplicație care permite utilizatorilor să comande mâncare pentru livrare sau preluare. În plus, un alt aspect notabil, este faptul că deși este o aplicație pentru rezervarea locurilor, platforma nu a fost concepută pentru rezervarea întregului restaurant pentru un anumit eveniment.

## 3. Grupurile de interese

- administratorii aplicației - comunică cu patronii de restaurante pentru a își extinde rețeaua de locații din aplicație; acest networking poate stârni apariția unor noi oportunități de business - publicitate reciprocă, parteneriate ș.a.
- clienții - cetățeni obișnuiți care doresc să ia masa în oraș și doresc să consulte un ghid al locațiilor din zonă, atât pentru a descoperi locații noi, a citi părerile autentice ale altor clienți, cât și a face rezervări.
- managerii de restaurant - aceștia beneficiază întrucât își expun afacerea unei audiențe largi pentru a atrage potențiali clienți și pot exploata potențialul de marketing și de dezvoltare al localurilor lor prin intermediul utilizării acestei platforme.

## 4. Colectarea cerințelor

### 4.1.Metode directe

Referințe către cerințele colectate în mod direct de la grupurile de interese (interviuri, chestionare etc) – dacă e cazul.

#### 4.1.1. Cerințele echipei de proiect

Pentru realizarea acestei aplicații s-a optat pentru integrare Web, deoarece are o acoperire mai mare pentru useri și poate fi distribuită mai ușor pentru vizibilitate. Mai mult, poate fi accesată de pe mai multe tipuri de dispozitive, dacă au acces la un browser.

Drept tehnologii alese pentru acest proiect, am optat pentru Spring (Java) pentru Backend și React (Typescript) pentru Frontend. Ambele tehnologii sunt populare în domeniul în care sunt folosite și pentru scopul realizării aplicației oferă suportul necesar pentru integrarea cerințelor prioritare.

## 5. Interpretarea cerințelor

**Cerințe funcționale**

- Sistemul trebuie sa ofere utilizatorului posibilitatea de a vizualiza restaurantele înscrise în aplicație atât sub formă de listă, cât și sub formă de pagini individuale, deoarece:
    - Informația despre restaurantele disponibile nu este centralizată, fiind nevoie de consultarea mai multor site-uri/forumuri.
    - Nu toate restaurantele au un site propriu, separat, unde pot fi găsite toate detaliile de care un client are nevoie atunci când face o alegere.
- Sistemul trebuie sa ofere posibilitatea utilizatorilor de a face rezervari la restaurantele din aplicație:
    - Clienții apreciază opțiunea de a putea efectua rezervări din același loc de unde se informează
    - Clientul se bucură de flexibilitate, putând efectua rezervări în afara programului de lucru al restaurantelor, fapt văzut ca un avantaj de către acest grup
    - Mesele dorite vor fi selectate de pe harta restaurantului, fapt ce oferă flexibilitate

**Cerințe de calitate**

Sistemul trebuie sa fie cat mai ușor de folosit, oferind flow-uri intuitive, care minimizează fricțiunea dintre client și aplicație, pentru a ușura procesul de selectare a restaurantului dorit, deci a îndeplinirii misiunii aplicației. (Utilizabilitate)

Un specialist în testare, un stakeholder, sau chiar un dezvoltator ar trebui să poată verifica corectitudinea output-ului, consistența în răspunsuri, performanța, securitatea și alte atribute funcționale sau non-funcționale ale sistemului. [12] (Testabilitate)

Un sistem extensibil este un sistem a cărui implementare a fost făcută luând în considerare posibilitatea de a fi ulterior dezvoltat cu un minim de efort necesar pentru implementarea extensiei. Nivelul de extensibilitatea al aplicației este evaluat prin costurile adăugării de funcționalități noi sau extinderii celor deja existente. (Extensibilitate)

În ceea ce privește performanța aplicației, s-a urmărit eficiența operațiilor necesare atât pe frontend, cât și pe backend, prin respectarea standardelor tehnologice. Pe partea de frontend, s-au aplicat strategii precum reducerea re-randărilor, gruparea eficientă a operațiilor și memorarea informațiilor calculabile sub anumite condiții. Pe partea de backend, s-a adoptat un stil modular de lucru, eliminând redundanța și asigurând un control mai mare asupra resurselor utilizate. Evaluarea fiecărui proces în funcție de necesitatea de memoizare sau stocare în cache/localStorage a fost un aspect important pentru optimizarea performanței. (Performanță)

**Constrângeri**

Din punct de vedere a fezabilitatii, dezvoltarea aplicatiei este constransa de resursa umana disponibila, dezvoltatorii avand atat experienta tehnica limitata, cat și resurse de timp scăzute (avand și alte proiecte în derulare în paralel).

O alta constrangere relevanta o reprezinta bugetul inexistent, fapt ce limitează dezvoltarea soluției software spre tehnologii open-source si servicii gratuite.

## 6. Prioritizarea cerințelor

**Cerințe funcționale**

1. Sistemul oferă utilizatorului posibilitatea de a vizualiza restaurantele înscrise în aplicație atât sub formă de listă, cât și sub formă de pagini individuale.
2. Clientul efectuează rezervari la restaurantele dorite prin intermediul platformei, pentru a realiza misiunea aplicației.
3. Pentru autentificarea clientului drept o persoană reală și pentru a combate rezervările false, ca manager de restaurant, aș dori ca clienții să confirme prin SMS sau email rezervările efectuate.
4. Sistemul permite clienților care își onorează rezervarea dreptul de a lăsa recenzii.
5. Utilizatorul își poate crea cont și se poate autentifica pe baza acestuia în aplicație.

**Cerințe de calitate**

1. Extensibilitate
2. Testabilitate
3. Performanță
4. Utilizabilitate

**Constrângeri**

1. Lipsa de buget și numărul mic de dezvoltatori.

**7. Specificații de analiză arhitecturală (selecție)**
Descompunerea lipseste. Exemple de US, UC, QS deasupra la curs 3.

# 14. Exemple din proiecte – Proiectare arhitecturala

1. si 2. Identic cu cel de mai sus

### 3. Tacticile arhitecturale

**QS3. Performanța:** Pentru îndeplinirea cerințelor de performanța vom aplica următoarele tactici arhitecturale:
- Vom controla fluxul de resurse: În cadrul implementării, atât pe BE și FE, vom face numai operații strict necesare pentru furnizarea unui răspuns, iar aceste operații vor folosi principii de codare agreate de tehnologiile folosite astfel încât operațiile să fie făcute cât mai eficient. Pe FE în special se va lua în considerare reducerea numărului de apeluri pentru reconstruirea elementelor, astfel încât impactul computațional pentru afișarea componentelor să fie minim.
- Vom gestiona resursele într-un mod constructiv, acolo unde este posibil vom încerca să păstrăm un comportament asincron asupra operațiilor executate, astfel încât să ne asigurăm că într-un anumit interval de timp ne folosim cât mai inteligent de resursele de care dispunem. În ceea ce privește datele de care dispunem, în cazul în care putem păstra în memorie anumite instanțe care ne ajută să procesăm mai repede unele operații, vom compara costurile de păstrare în memorie cu potențialele costuri dacă facem operațiile fără a integra acest aspect și vom alege varianta mai eficientă.
- În cazul asincron menționat mai sus, va trebui să avem în vedere modul în care gestionăm prioritățile de execuție a operațiilor. Vom încerca să menținem un sistem în care operațiile non prioritare nu vor bloca stiva de apeluri.

**QS4. Utilizabilitate:** Tacticile arhitecturale folosite în asigurarea îndeplinirii cerintei de calitate Utilizabilitate in cadrul dezvoltării aplicației vor fi: - La design time:
- Incremental design process: Pe parcursul întregii dezvoltări se vor aplica tehnici de iterare succesiva si prototipare pentru a genera interfața potrivită aplicației
- La run time:
  - Feedback constant: - Vom implementa un sistem de notificări de tip toast (mesaje scurte care apar în colțul paginii) care va atentiona utilizatorii atunci cand este nevoie.
  - Vom folosi componente vizuale(butoane, drawers etc) cu animatiile specifice paradigmelor de design modern, a.i. utilizatorul sa poata avea o intelegere intuitiva asupra modului in care se interactioneaza cu aplicatia.
  - Responsive design: Vom apela atât la biblioteci care usureaza lucrul cu pagini ce trebuie randate pe mai multe platforme (precum ChakraUI), cât și la proprietăți ale CSS-ului care permit randarea diferită în funcție de criterii precum rezoluția ecranului utilizatorului.

### 4. Specificațiile de proiectare

**QS4. Utilizabilitate**

Din perspectiva tacticilor arhitecturale folosite pentru a asigura îndeplinirea cerintei Utilizabilitate, se remarca următoarele aspecte din user story-urile stabilite în raportul arhitectural:

- US2. Întregul flow de creare a rezervarii la un restaurant, integral pentru succesul platformei, trebuie sa fie cat mai ușor de folosit posibil. Pentru a îndeplini acest lucru, atenție deosebită trebuie acordată utilizabilitatii. Asadar:

x Interfata trebuie sa ghideze utilizatorul în timpul întregului proces. Vom apela la o solutie tip “step wizard”, un pattern folosit des în aplicațiile prezentului.

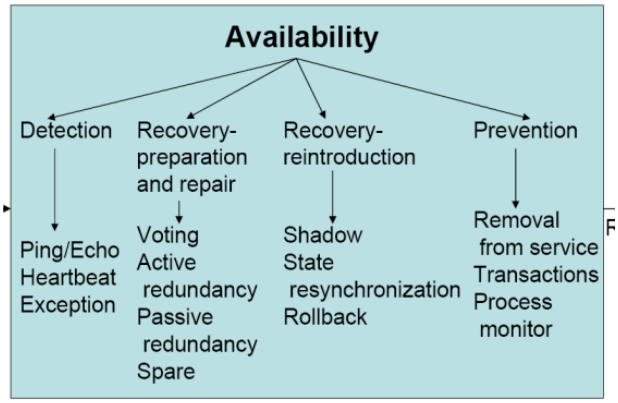
X Interfata trebuie sa semnaleze eventualele inadvertente în completarea formularului de rezervari si sa notifice utilizatorul cu mesaje utile, care sa ajute la remedierea problemei.

x Harta pentru selectarea locurilor trebuie sa fie clar de urmărit, să distingă între mesele disponibile, mesele ocupate și mesele aflate în selecția curentă a utilizatorului.

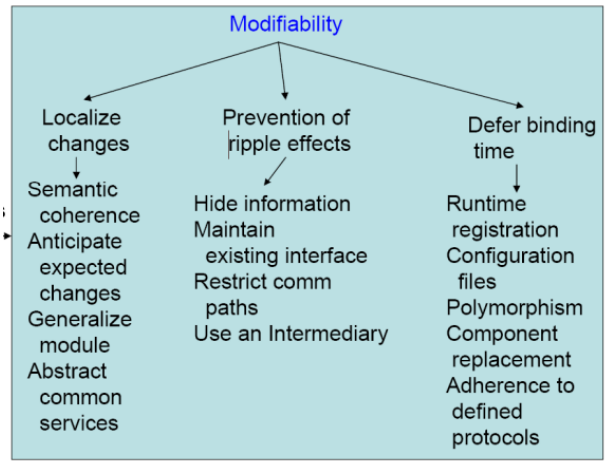
- US1. Plecând tot de la ideea familiaritatii, în implementare propunem o soluție care se bazează pe componenta de design “Card”, popularizata de catre Material Design. Asadar, ne dorim ca lista restaurantelor sa se înfățișează sub forma unei liste de elemente tip Card, ce contin o imagine, numele restaurantului, rating si o scurta descriere a sa. Apasarea pe Card va deschide pagina detaliată a restaurantului.

# 15. Anexa – Strategii + Tactici in relatie cu atributele de calitate

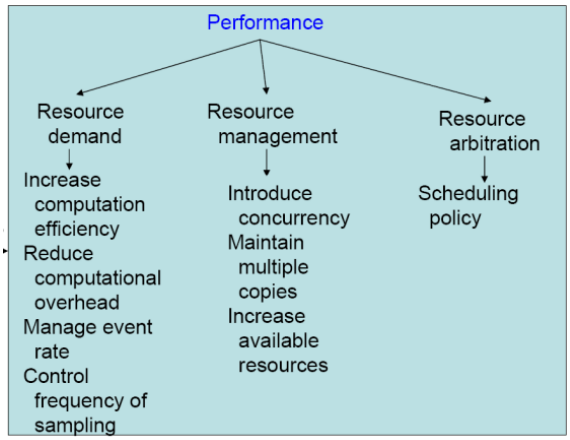
| | |
|---|---|
| Source | Internal/external to system |
| Stimulus | Fault: omission, crash, timing, response etc |
| Artifact | System's processors, communication channels, persistent storage, processes etc |
| Environment | Normal operation or degraded mode |
| Response | Detect event and record it/notify appropriate parties/disable event sources causing faults/failures/ be unavailable for an interval/ continue |
| Response measure | Time interval of available system, availability time, time interval of degraded mode, repair time |



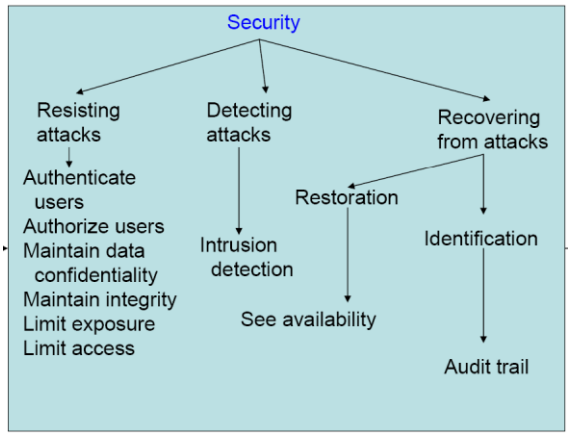

| | |
|---|---|
| Source | End user, developer, system administrator |
| Stimulus | Wishes to add/delete/modify/vary functionality, QA, capacity etc |
| Artifact | System UI, platform, environment, systems that interoperates with target system |
| Environment | Design time, implementation integration , build time, runtime |
| Response | Locates place in architecture to modify, makes modification w/o affecting other functions, tests modifications, deploys modifications |
| Response measure | Costs in terms of number of elements affected, effort, money; extent to which this affects other QAs, functions |




| | |
|---|---|
| Source | Independent sources |
| Stimulus | Periodic or stochastic or sporadic events occur |
| Artifact | System |
| Environment | Normal mode, overload mode |
| Response | Processes stimuli; changes level of service |
| Response measure | Latency, deadline, throughput, jitter, miss rate, data loss |



| | |
|---|---|
| Source | Individual/system: identity, internal/external, authorization, access |
| Stimulus | Try to: display data, change/delete data, access system services, reduce availability |
| Artifact | System services, data within system |
| Environment | On/offline, (dis)connected, firewalled or open |
| Response | Various |
| Response measure | Various |



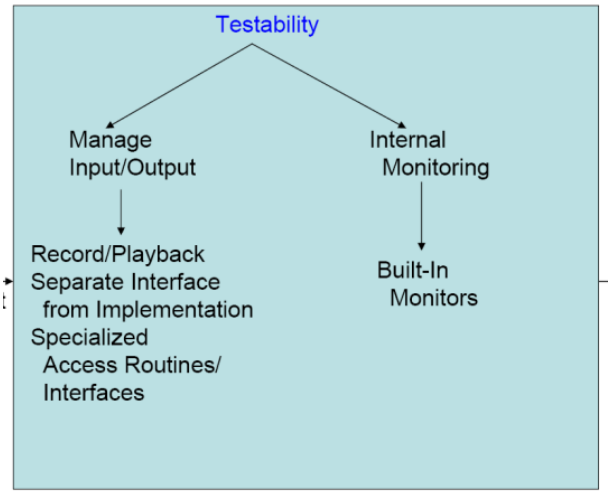

## Security Scenario Responses

- Authenticates user
- Hides identity of user
- Blocks/allows acces to data/services
- Grants/withdraws permission to access data/services
- Records access/modification or attempts
- Stores data in certain formats
- Recognizes access/usage roles
- Informs users on other systems
- Restricts availability of services

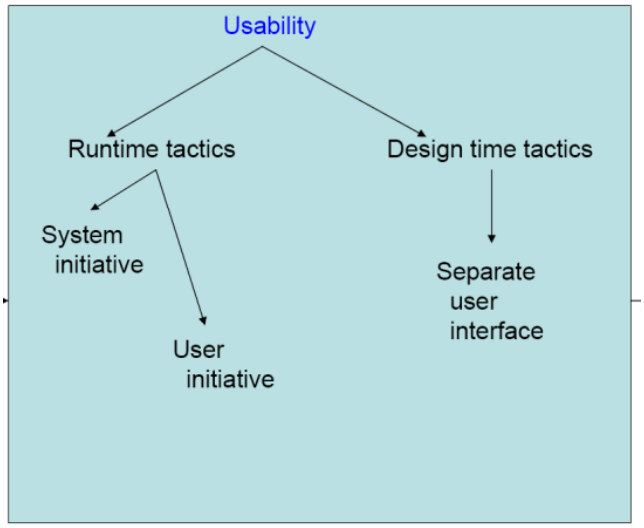
## Security Scenario Response Measure

- Time/effort/resources for circumventing security measures successfully
- Probability of detecting attack, identifying attacker
- Percentage of services still available under DoS attack
- Restore data/services
- Extent to which data/services damaged or legitimate access denied

| | |
|---|---|
| Source | Unit developer, increment integrator, system verifier, client acceptance tester, system user |
| Stimulus | Analysis, architecture, design, class, subsystem integration completed, system delivered |
| Artifact | Design part, code part, complete application |
| Environment | At design time, at development time, at compile time, at deployment time |
| Response | Provides access to state values; provides computed values; prepares test environment |
| Response measure | Percent executable statements executed, probability of failure if fault exists, time to perform tests, length of longest dependency chain in a test, length of time to prepare test environment |



| | |
|---|---|
| Source | End user |
| Stimulus | Wants to learn system features, use system efficiently, minimize impact of errors, adapt system, feel comfortable |
| Artifact | System |
| Environment | At runtime and configure time |
| Response | Various |
| Response measure | Task time, number of errors, number of problems solved, user satisfaction, user knowledge gain, ratio of successful operations to total operations, amount of time/data lost |




# Usability Scenario Responses

- System provides responses to support
  - Learn system features
  - Use system efficiently
  - Minimize impact of errors
  - Adapt system
  - Feel comfortable