
Analiza și proiectarea sistemelor software

Curs 9

PLAN CURS

- Arhitectura și specificul sistemelor și aplicațiilor Web

Analiza aplicațiilor Web

- Modelul analiză pentru aplicații Web
- Analiza relații-navigare

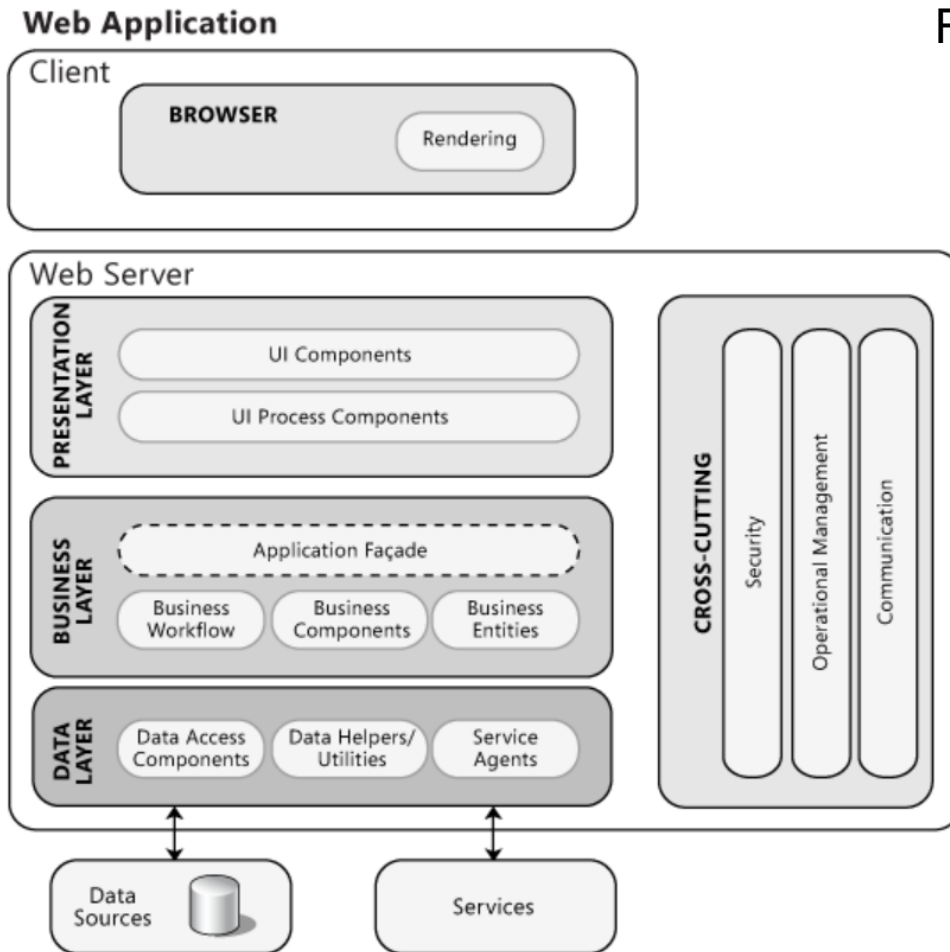
Proiectarea aplicațiilor Web

- Principii
- Modelul proiect pentru aplicații Web

SPECIFICUL SISTEMELOR ȘI APLICAȚIILOR WEB

- Utilizarea intensivă a rețelelor
- Concurență : utilizatori multipli simultan
- Încărcare nepredictibilă
- Performanță – acces simplu și rapid la informații
- Disponibilitate – permanent
- Determinate de date –
 - utilizarea hypermedia pentru a prezenta diferite tipuri de conținut (text, grafic, video).
 - acces la baze de date și la depozite de date.
- Sensibile la conținut – calitatea aplicației e determinată de calitatea conținutului prezentat
- Evoluție continuă – conținut dinamic \Rightarrow necesitatea unei infrastructuri pentru evoluția și explorarea continuă a conținutului.
- Cerința de dezvoltare și lansare rapidă pe piață
- Securitate – protejare conținut, transfer de date protejat și sigur \Rightarrow măsuri puternice de securitate implementate atât în infrastructură cât și la nivelul aplicației.
- Estetică – caracteristică fundamentală pentru calitatea aplicației.

STRUCTURA TIPICĂ A APLICAȚIILOR WEB



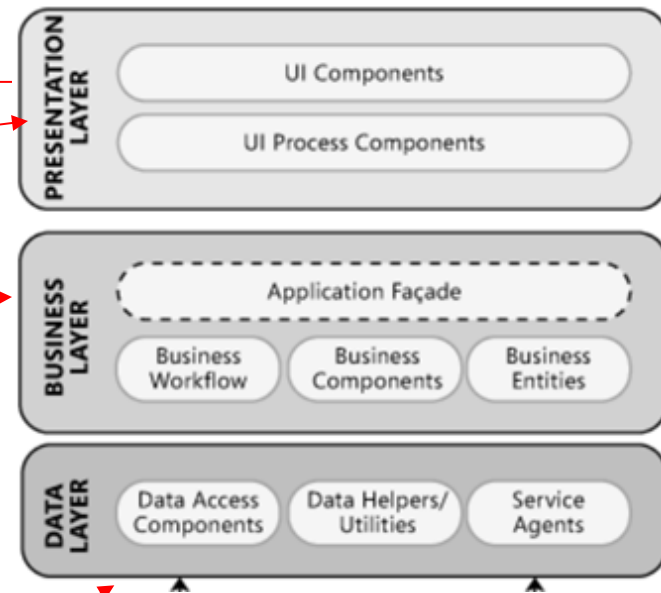
Respectă principii generale de proiectare.

- Divizarea logică a aplicației pe mai multe nivele.
- Utilizarea de abstractizări pentru cuplare slabă între straturi.
 - componente de tip facade cu interfețe bine cunoscute (ascund particularitățile altor componente)
 - interfețe
 - clase abstracte

NIVELE TIPICE ALE APLICAȚIILOR WEB

- Separare componente UI (preferabil standadizate) de componente ale logicii de prezentare.

- Separare componente, entități și procese business.
- Entitățile business reprezintă date din lumea reală.
- Nivel stateless \Rightarrow reduce competiția pentru resurse, crește performanța.
- Interfață bazată pe mesaje.
- Utilizare tranzacții pentru operații critice \Rightarrow menținerea integrității și prevenirea pierderii de date.



- Abstractizează logica de acces la date \Rightarrow configurare, mentenanță, ascunde detalii acces BD.
- Proiectare obiecte entitate pe care le poate popula acest nivel sau folosite pentru actualizarea sursei de date, utilizare DTO (data transfer objects) pentru transfer date între nivele.
- *Connection pooling* (acumulatori de conexiuni)
- Serializare operații \Rightarrow reducerea numărului de accesuri la BD
- Tratare erori acces date, propagare excepții către nivelul business.

ANALIZA ȘI PROIECTAREA APLICAȚIILOR WEB

Analiza : modele

Cerințe funcționale

- cazuri de utilizare
- procese

Conținut

- model domeniu

Proiectarea : modele

Aspecte informaționale

- model conținut

Structură hypertext și funcții de navigare

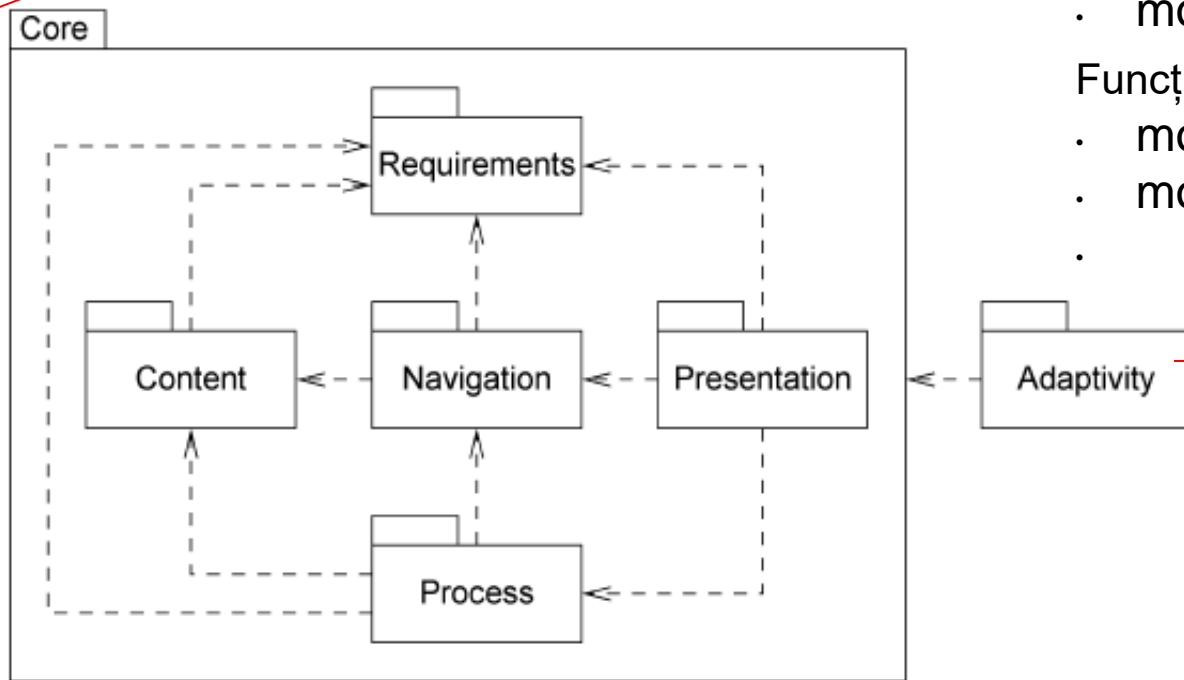
- model navigare

Schema layout

- model prezentare

Funcționalitate

- model proces
- model adaptivitate
-

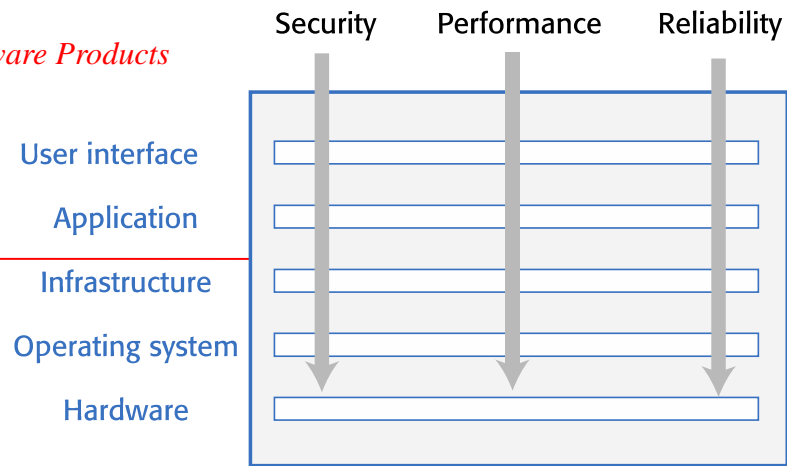


Conținut : inserare/eliminare caracteristici text/multimedia, variante de conținut, etc.

Navigare : ordonare, adnotare, ascundere, generare link-uri.

Prezentare : audio/video, text, limbă, variante de layout (redimensionare font, imagini, schimbare culori), ...

PREOCUPĂRI TRANSVERSALE: SECURITATE



Atacatorii pot încerca să folosească vulnerabilități ale diferitelor tehnologii utilizate pe straturi diferite (ex. SQL database, Firefox browser).



Protecție la atacuri pe fiecare strat: straturile inferioare protejate de atacurile care au trecut cu succes de straturile superioare.

Variante :

- Componentă unică pentru securitate = vulnerabilitate critică a sistemului dacă își întrerupe funcționarea corectă sau este compromisă de un atac.
- Securitate distribuită pe straturi \Rightarrow sistem mai rezilient la atacuri și la defectare software.

FUNCȚIONALITATE MULTI-STRATIFICATĂ ÎN APLICAȚIE BAZATĂ WEB

Perspectiva statică

Interfață sistem în browser Web (HTML, CSS, JavaScript).
Interfață Mobile implementată ca app.

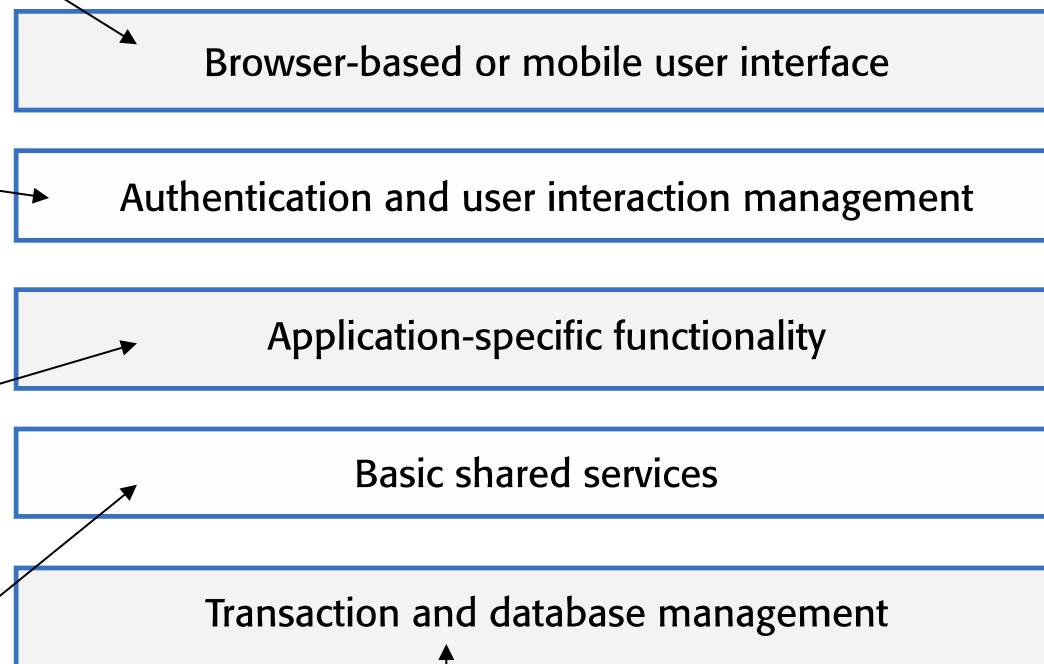
Figure 4.10 A generic layered architecture for a web-based application

Ian Sommerville – Engineering Software Products

Strat administrare UI:
autentificare utilizatori,
generare pagini web.

Funcționalitatea aplicației.
Se poate expanda pe mai
multe straturi.

Furnizează servicii folosite
de stratul cu
funcționalitatea aplicației.



Furnizează servicii pentru administrarea bazei de
date, administrarea tranzacțiilor, recuperare.

COMUNICAREA CLIENT-SERVER

Perspectiva dinamică

Comunicarea client-server web folosește frecvent protocolul HTTP.

Clientul trimite la server un mesaj care include o *instrucțiune* ca GET sau POST și un *identificator al unei resurse* (de obicei un URL) pe care instrucțiunea respectivă trebuie să opereze. Mesajul poate include și *informații adiționale*, ca cele colectate dintr-un form HTML.

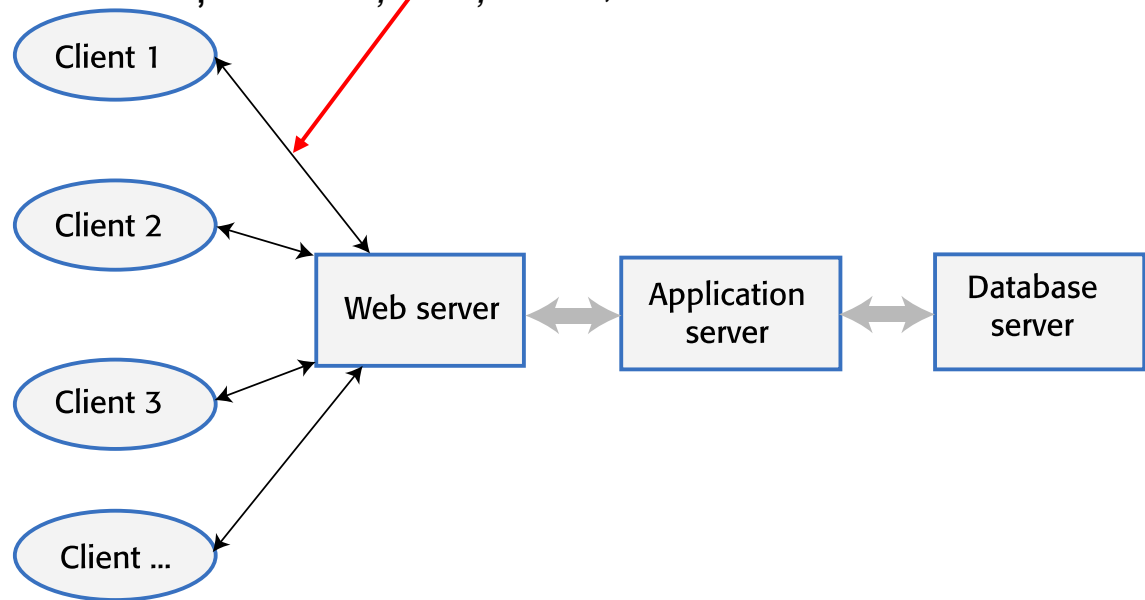


Figure 4.14 **Multi-tier** client-server architectural model

Ian Sommerville – *Engineering Software Products*

Moduri de reprezentare date, în format text structurat, transferate prin protocolul HTTP:

- XML – limbaj de marcare cu etichete folosite pentru a identifica fiecare element de date.
- JSON – reprezentare simplă bazată pe reprezentarea obiectelor în limbajul JavaScript.

DECIZII pentru TEHNOLOGII FOLOSITE

Bază de date - Tipuri de baze de date folosite frecvent:

- Relaționale - date organizate în tabele structurate

Dacă este necesar managementul tranzacțiilor și când structurile datelor sunt predictibile și destul de simple.

- NoSQL – datele au o organizare flexibilă, definită de dezvoltator.

Mai flexibile și potențial mai eficiente pentru analiza datelor (big data processing)

Server - Decizie cheie : rulare pe *servere la client* sau în *cloud*.

Recomandare :

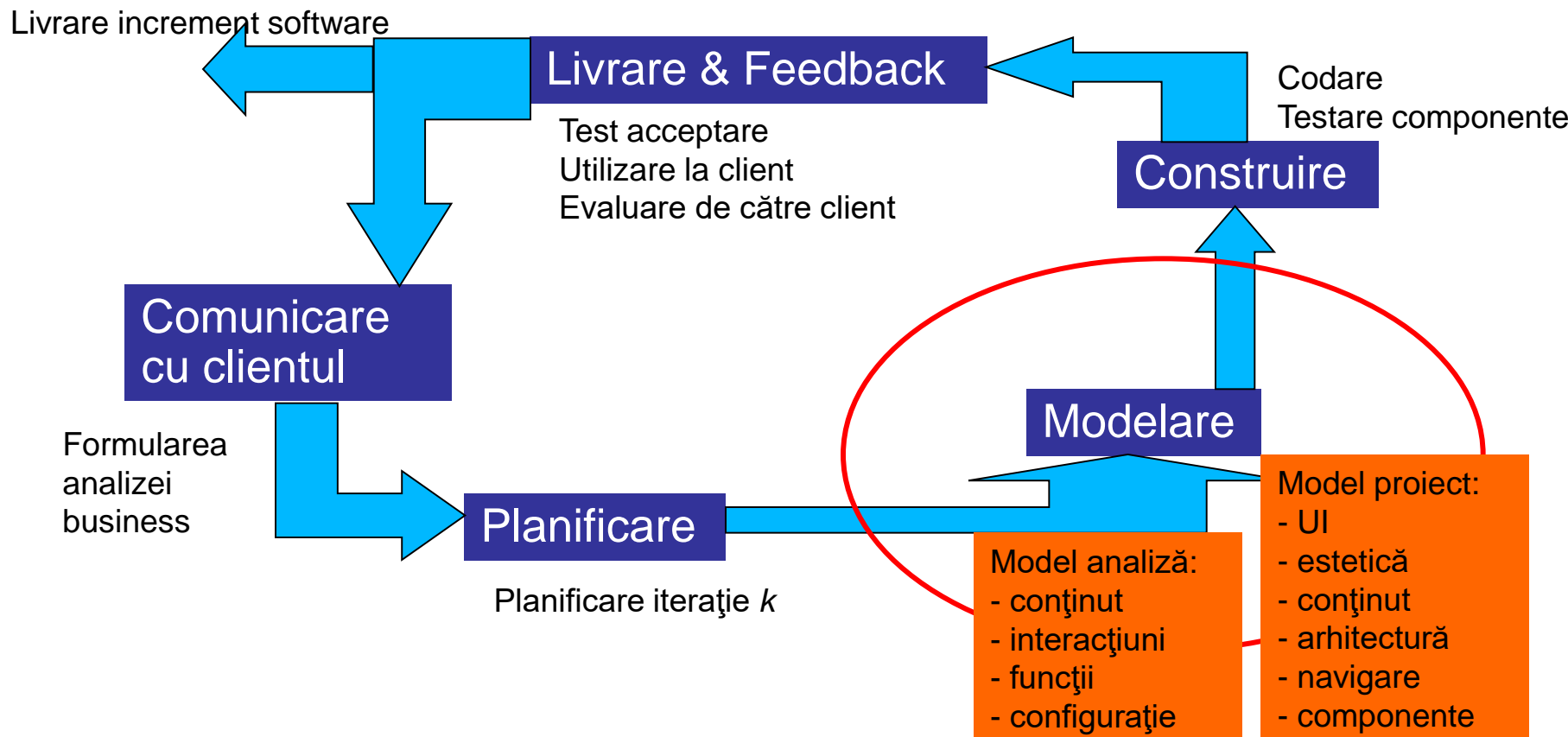
- Produsele destinate consumatorilor și care nu sunt simple aplicații mobile - cloud.
- Produse business : decizie dificilă : securitate vs. șablon de utilizare nepredictibil.

Framework-uri

- Avantaj : reutilizare \Rightarrow reducere costuri de dezvoltare și timp de lansare pe piață.
- Dezavantaje : constrângeri suplimentare și lipsa controlului asupra evoluției.

PROCESUL WebE (Inginerie Web)

PROCESUL WebE (inginerie Web)



RECOMANDĂRI PRACTICE PENTRU DEZVOLTAREA APLICAȚIILOR WEB

- Alocarea de timp suficient pentru înțelegerea *domeniului aplicației* și a *obiectivelor produsului*, în contextul unor formulări inițiale posibil vagi.
- Utilizarea unei abordări bazată pe *scenarii* (cazuri de utilizare) pentru descrierea modului de interacțiune a utilizatorilor cu aplicația.
- Dezvoltarea unui *plan*, chiar și foarte succint, al proiectului (în contextul cerinței de lansare rapidă a aplicației, granularitatea planului va fi mică (ex. zilnic)).
- Realizarea modelelor *analiză* și *proiect* ale aplicației; nu excesiv de cuprinzătoare; diagrame de secvențe, de clase, de stare.
- Revizuirea *consistenței* și *calității* modelelor (revizuri tehnice formale)
- Utilizarea *instrumentelor* și *tehnologiilor* ce permit construirea sistemului folosind un număr cât mai mare de *componente reutilizabile*.
- Planificarea de *teste exhaustive* și executarea lor *înainte* de lansarea sistemului. (“șansă unică”, “test first then deploy”).

RECOMANDĂRI PRACTICE PENTRU DEZVOLTAREA APLICAȚIILOR WEB

RECOMANDĂRI PENTRU CALITATEA APLICAȚIILOR WEB

www.webstyleguide.com/wsg3/index.html

www.w3.org/Provider/Style

www.worldbestwebsites.com

www.killersites.com/core.html

www.useit.com

www.pantos.org/atw

www.asktog.com

PLAN CURS

- Arhitectura și specificul sistemelor și aplicațiilor Web

Analiza aplicațiilor Web

- Modelul analiză pentru aplicații Web
- Analiza relații-navigare

Proiectarea aplicațiilor Web

- Principii
- Modelul proiect pentru aplicații Web

ANALIZA CERINȚELOR APLICAȚIILOR WEB

Etapa de formulare

- Identificarea *obiectivelor* aplicației Web
- Definirea *categoriilor* de utilizatori

Etapa de culegere

- *Comunicare* între membrii echipei de dezvoltare și stakeholder-i (clienți, utilizatori finali)
- Stabilirea *ierarhiei* utilizatorilor (categorii și sub-categorii de utilizatori)
- Listarea cerințelor de *conținut* și de *funcționalitate*
- Definirea *scenariilor de interacțiune* (**cazuri de utilizare**) d.p.d.v. al utilizatorilor finali.

MODELUL ANALIZĂ PENTRU APLICAȚII WEB

MODELUL ANALIZĂ - determinat de descrierea *cazurilor de utilizare*
⇒ *conținutul* informațional și *funcționalitatea*.

METODĂ

Colectare informații

Revizuire informații, realizare modificări necesare

Realizarea modelului analiză.

Activitățile analizei:

- Analiza conținutului
- Analiza interacțiunilor
- Analiza funcționalității
 - funcțiile de procesare a conținutului
 - funcțiile independente de conținut
- Analiza configurării: contextul și infrastructura

Obs. Rigurozitatea definirii cazurilor de utilizare și a modelării trebuie să crească odată cu complexitatea aplicației.

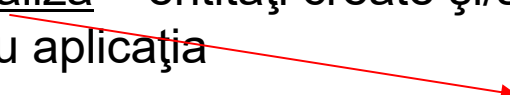
MODELUL CONȚINUTULUI

MODELUL CONȚINUTULUI

Determinat prin examinarea cazurilor de utilizare !

Elementele modelului:

- Elemente structurale compuse din obiecte conținut
(ex. obiecte: text, grafice, foto, video, audio)
- Clasele de analiză – entități create și/sau manipulate în timpul interacțiunii utilizatorului cu aplicația



Definite prin attribute, operații și colaborări.
Reprezentate în diagramă de clase.

MODELUL CONȚINUTULUI

Modelul conținutului
Modelul interacțiunilor
Modelul funcțional
Modelul configurației

DEFINIREA OBIECTELOR CONȚINUT

Conținut

- format din informații preexistente
- dezvoltat independent de dezvoltarea aplicației (înainte, în paralel sau după)
- incorporat în structura de ansamblu a aplicației prin referire navigațională
- dezvoltat de persoane specializate (nu de ingineri software)

Obiect de tip conținut = element coeziv cu informații ce urmează a fi prezentate utilizatorului final.

RELAȚII ȘI IERARHII ALE OBIECTELOR CONȚINUT

Variante de modelare conținut – funcție de complexitatea acestuia:

- Listă de obiecte cu descrieri ale acestora
- Diagrame ER
- Structură arborescentă (data tree) – ierarhie de obiecte conținut. (DOM)

MODELUL INTERACȚIUNILOR

Modelul conținutului
Modelul interacțiunilor
Modelul funcțional
Modelul configurației

CAZURI DE UTILIZARE

Elementul dominant al modelului interacțiunilor.

Componentele modelului:

- Cazuri de utilizare
- Diagrame de secvențe
- Diagrame de stări
- Prototipul interfeței utilizator

DIAGrame DE SECVENȚE

Reprezentare a modului în care acțiunile utilizatorului *colaborează* cu clasele de analiză, pentru fiecare caz de utilizare.

Axa orizontală : clasele de analiză folosite în cazul de utilizare.

Axa verticală : acțiunile definite în cazul de utilizare.

DIAGrame DE STĂRI ȘI TRANZIȚII (SMD)

Reprezentare comportament pe parcursul desfășurării interacțiunii, sub formă de stări și tranziții între stări.

Utilitate: relevă informații despre căi de navigare posibile ce nu au fost identificate în cazurile de utilizare sau cu diagramele de secvențe.

Reprezintă dimensiuni complementare, permițând depistarea omisiunilor și inconsistențelor.

MODELUL INTERACȚIUNILOR

Modelul conținutului
Modelul interacțiunilor
Modelul funcțional
Modelul configurației

PROTOTIPUL INTERFEȚEI UTILIZATOR

Elemente:

- Layout-ul (așezarea în pagină)
- Conținutul prezentat
- Mecanismele de interacțiune
- Elementele de estetică

Componentele modelului:

- Cazuri de utilizare
- Diagrame de secvențe
- Diagrame de stări
- Prototipul interfeței utilizator

Trebuie să implementeze legăturile majore de *navigare* și să reprezinte *layout*-ul general al ecranului.

Realizare:

- Manuală
- Utilizând instrumente software – recomandabil, deoarece facilitează vizualizarea fluxului navigațional.

MODELUL FUNCȚIONAL

Adresează 2 elemente de procesare ale aplicației Web:

- Funcționalitatea observabilă de către utilizatori

Nivel ridicat de abstractizare procedurală

Funcții de procesare inițiate direct de utilizator.

- Operațiunile conținute în clasele de analiză, care implementează comportamentele asociate fiecărei clase.

Nivel scăzut de abstractizare procedurală

Funcții de procesare ce trebuie implementate în operațiunile claselor.

Manipulează atributele clasei și sunt implicate în colaborările clasei.

Reprezentare cu diagrame de activitate : ilustrează fluxul de procesare și deciziile logice de pe parcursul acestuia.

MODELUL CONFIGURAȚIEI

Aplicațiile Web trebuie proiectate și implementate a.î. să permită instalarea lor pe o *varietate de contexte* atât de partea *clientului* cât și de partea *serverului*.

Modelul configurației pentru *aplicații simple* = listă de atribute de partea serverului și de partea clientului:

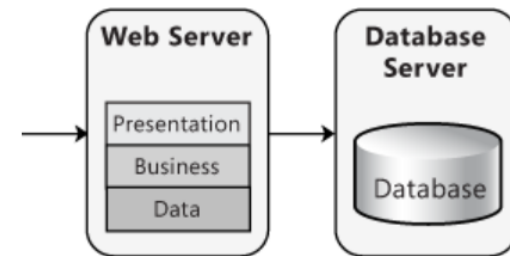
- *Platforma* hardware & sistem de operare.
- *Interoperabilitate* de partea *serverului* : interfețe corespunzătoare, protocoale de comunicare, informații de colaborare.
- *Compatibilitate* de partea *clientului* \Rightarrow necesitatea testării exhaustive a aplicației cu toate tipurile de browser-e specificate în modelul configurației.

Modelul configurației pentru *aplicații complexe*:

- *Distribuire* pe mai multe servere
- Arhitecturi pentru *caching*
- Baze de date la distanță
- *Servere multiple* pentru diferite obiecte de pe o *pagină*.

Reprezentare utilizând diagrama UML de instalare (deployment).

VARIANTE DE INSTALARE (deployment)

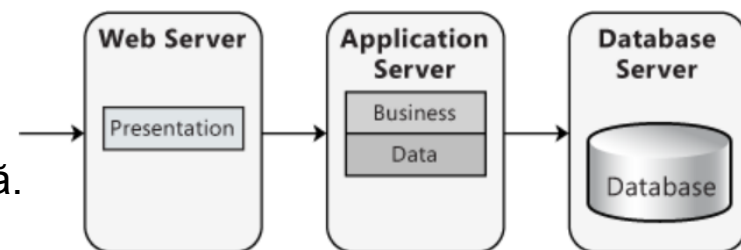


Nedistribuită – favorizează performanța

- Dacă performanța e critică și logica business nu trebuie partajată cu alte aplicații.
- Nu e necesară autentificare separată dacă logica de prezentare și cea business sunt în același proces.
- E necesară o metodă de transfer date sensibile între serverul web și serverul BD.

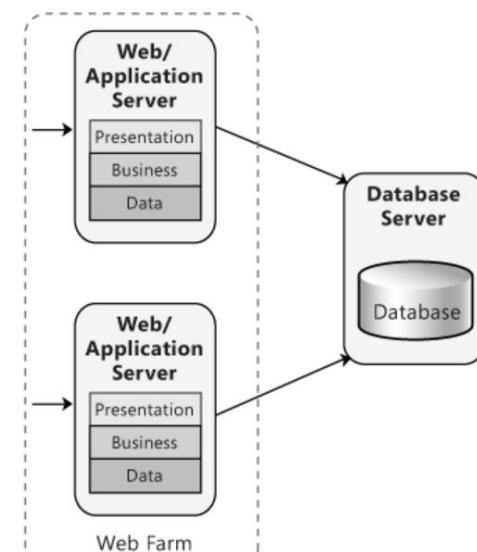
Distribuită – favorizează scalabilitatea și securizarea separată a nivelelor

- Dacă scalabilitatea și/sau securizarea separată primează.
- Interfață bazată pe mesaje pentru nivelul business
- Performanță cu comunicare prin TCP cu codificare binară
- E necesară protejarea datelor sensibile transferate între treptele fizice



Instalare pe servere multiple. Echilibrarea încărcării.

- Distribuie cereri \Rightarrow minimizare timp răspuns, optimizare utilizare resurse, creștere debit (rată de transfer).
- Componente stateless, utilizare serviciu de stare *out-of-process* sau serviciu de BD.
- Utilizare servicii specializate pentru echilibrarea încărcării.
- Partiționare, pe mai multe servere, a BD cu cerințe mari de acces.



PLAN CURS

- Arhitectura și specificul sistemelor și aplicațiilor Web

Analiza aplicațiilor Web

- Modelul analiză pentru aplicații Web
- Analiza relații-navigare

Proiectarea aplicațiilor Web

- Principii
- Modelul proiect pentru aplicații Web

ANALIZA RELAȚII-NAVIGARE (RNA)

În contextul aplicațiilor Web, fiecare element arhitectural are *potențial de conectare* cu alte elemente arhitecturale.

Număr mare de legături \Rightarrow complexitate crescută a navigării.

Obiectiv : stabilirea de legături între obiecte conținut și între funcții pentru a oferi capacitățile solicitate de utilizator.

RNA - tehnică sistematică de determinare a

Implementate prin mecanisme de navigare (ex. link, buton, menu)

- ***structurii de relații*** utile unei aplicații,
- ***structurilor de navigare*** construite peste structura de relații.

În etapele analizei relații-navigare se urmărește și identificarea relațiilor între elemente neacoperite în etapele anterioare de crearea a modelului analiză.

ANALIZA RELAȚII-NAVIGARE

Etapele RNA

- Analiza stakeholder-ilor
 - Identificarea categoriilor de utilizatori,
 - Stabilirea unei ierarhii a acestora.
- Analiza elementelor
 - Identificarea obiectelor conținut,
 - Identificarea elementelor funcționale.
- Analiza relaționărilor
 - Definirea relațiilor dintre elemente.
- Analiza navigării
 - Examinarea modalităților de acces al utilizatorilor la elemente sau grupuri de elemente.
- Analiza de evaluare
 - Evaluări (ex. cost / beneficiu) asociate cu implementarea relaționărilor.

ANALIZA RELAȚII-NAVIGARE

ANALIZA RELAȚIONĂRILOR - Definirea *relațiilor dintre elemente*.

Listă de întrebări aplicate *fiecărui element* (obiect de date sau funcție) determinat în etapele anterioare.

1. Elementul este membru al unei categorii mai largi de elemente ?
2. Care sunt atributele sau parametrii identificați pentru element ?
3. Există informații descriptive pentru element ? Unde se află acestea ?
4. Elementul apare în locații diferite în cadrul aplicației ? Care sunt aceste locații ?
5. Elementul este compus din subelemente ? Care sunt acestea ?
6. Elementul este membru al unei colecții de elemente ? Care este aceasta și care este structura ei ?
7. Elementul este descris de o clasă de analiză ?
8. Există elemente similare cu cel considerat ? Se pot combina acestea într-un singur element ?
9. Elementul este utilizat într-o ordonare specifică a altor elemente ? Apariția sa depinde de alte elemente ?
10. Există un element ce apare întotdeauna după elementul analizat ?
11. Care sunt pre- și post-condițiile ce trebuie îndeplinite pentru utilizarea elementului ?
12. Elementul este folosit de categorii specifice de utilizatori ? Categorii diferite de utilizatori îl folosesc în moduri diferite ? Cum anume ?
13. Se poate asocia elementul cu o anumită formulare a unui obiectiv ?
14. Apare întotdeauna simultan cu alte elemente ? Care sunt acestea ?
15. Apare mereu în același loc (pe ecran sau în pagină) cu alte elemente ? Care sunt acestea ?

ANALIZA RELAȚII-NAVIGARE

ANALIZA NAVIGĂRII - Examinarea *modalităților de acces* al utilizatorilor *la elemente* sau grupuri de elemente.

Listă de întrebări:

1. Este necesar ca anumite elemente să fie mai simplu de accesat decât altele ? Care este prioritatea prezentării ?
2. Este necesar ca anumite elemente să fie evidențiate, a.î. utilizatorul să fie forțat să navigheze către ele?
3. Cum vor fi gestionate erorile de navigare ?
4. Navigarea către grupuri de elemente trebuie să fie prioritară în raport cu navigarea către un anumit element ?
5. Care este modalitatea de realizare a navigării : link-uri, acces bazat pe căutare, altele ?
6. Prezentarea anumitor elemente trebuie făcută pe baza contextului acțiunilor anterioare de navigare?
7. Este necesară menținerea unui jurnal (log) al navigării ?
8. Trebuie să fie disponibilă o hartă completă a navigării în fiecare punct al interacțiunii cu utilizatorul sau sunt suficiente link de tip “back” sau pointer de direcționare ?
9. Proiectarea navigării este determinată de majoritatea comportamentelor comune ale utilizatorului sau de importanța percepută a elementelor definite pentru aplicație ?
10. Poate utilizatorul reține navigarea anterioară în vederea reluării ei ?
11. Pentru ce categorie de utilizatori trebuie proiectată navigare optimă ?
12. Cum trebuie manipulate link-urile externe aplicației ? Suprapuse peste browser-ul existent ? Într-o nouă fereastră a browser-ului ? Ca un frame separat ?

ANALIZA RELAȚII-NAVIGARE

ANALIZA NAVIGĂRII - cerințe generale

Exemple de întrebări cheie:

- Este necesară prezentarea hărții site-lui (site map) care să ofere o viziune de ansamblu a structurii acestuia ?
- Va fi disponibil un tur ghidat al utilizatorului care să evidențieze elementele (obiecte conținut și funcții) disponibile cele mai importante ?
- Va putea utilizatorul accesa elemente pe baza atributelor acestora ? (ex. toate fotografiile unei anumite clădiri, toate funcțiile ce permit calcularea greutateii, etc.)

PLAN CURS

- Arhitectura și specificul sistemelor și aplicațiilor Web

Analiza aplicațiilor Web

- Modelul analiză pentru aplicații Web
- Analiza relații-navigare

Proiectarea aplicațiilor Web

- Principii
- Modelul proiect pentru aplicații Web

To some, Web design focuses on visual look and feel...

To others, Web design is about structuring information and navigation through the document space.

Others might even consider Web design to be about the technology used to build interactive Web applications.

In reality, design includes all of these things and maybe more.

Thomas Powell

PROIECTAREA APLICAȚIILOR WEB – PRINCIPII

Proiectarea aplicațiilor Web include:

- Activități tehnice
 - Structura tehnică a aplicației – parte a proiectării arhitecturale și de navigare.
- Activități non-tehnice
 - Look&feel al conținutului – parte a proiectării graficii
 - Estetica layoutului – parte a proiectării interfeței UI

Principalele categorii de dezvoltatori:

- ingineri software
- designer-i ai graficii
- dezvoltatorii de conținut.

PROIECTAREA APLICAȚIILOR WEB – PRINCIPII DESIGN-ul și CALITATEA aplicațiilor Web:

- Utilizabilitate
 - Inteligibilitatea site-lui
 - Feedback și help on-line
 - Calități de interfață și estetice
 - Caracteristici speciale legate de logica aplicației
- Funcționalitate
 - Capabilități de căutare și de extragere informații
 - Navigare și explorare
 - Trăsături relative la domeniul aplicației
- Fiabilitate
 - Procesarea corectă a link-urilor
 - Recuperarea din eroare
 - Validarea și recuperarea intrărilor de la utilizator
- Eficiență
 - Timpul de răspuns
 - Viteza de generare a paginilor
 - Viteza de generare a graficii

PROIECTAREA APLICAȚIILOR WEB – PRINCIPII

DESIGN-ul și CALITATEA aplicațiilor Web:

- **Mentenabilitate**
 - Ușurința de a corecta
 - Adaptabilitate
 - Extensibilitate
- **Securitate**
 - Abilitatea de a riposta la accesele neautorizate
 - Abilitatea de a deturna atacurile malefice
- **Disponibilitate**
 - Permanent
 - În raport cu tipuri multiple de browser-e
- **Scalabilitate**
 - Număr de utilizatori
 - Volum de date
- **Timpul de lansare pe piață** - măsură a calității din punct de vedere al afacerii.

PROIECTAREA APLICAȚIILOR WEB – PRINCIPII

DESIGN-ul și CALITATEA aplicațiilor Web: listă de întrebări pentru **CALITATEA APLICAȚIEI**.

1. Pot fi conținutul, funcțiile, navigarea adaptate conform preferințelor utilizatorilor ?
2. Conținutul și funcționalitatea pot fi adaptate la lărgimea de bandă folosită de utilizator?
3. Grafica și alte media non-text sunt utilizate corespunzător ? Dimensiunile fișierelor grafice sunt optimizate pentru eficientizarea afișării ?
4. Tabelele sunt organizate și dimensionate pentru a fi inteligibile și afișate eficient ?
5. Codul (X)HTML este optimizat (pentru eliminarea ineficiențelor) ?
6. Design-ul general al paginii este ușor de “citit” și de navigat ?
7. Se trimit toate link-urile la informații de interes pentru utilizatori ?
8. Care este probabilitatea ca majoritatea link-urilor folosite să fie persistente în Web ?
9. Este aplicația Web instrumentată cu utilități de management ce includ instrumente pentru urmărirea utilizării, testarea link-urilor, căutare locală și securitate ?

PROIECTAREA APLICAȚIILOR WEB – PRINCIPII

DESIGN-ul și CALITATEA aplicațiilor Web: listă de întrebări pentru **CALITATEA CONȚINUTULUI**.

1. Domeniul și profunzimea conținutului sunt conforme cu necesitățile utilizatorului ?
2. Se pot identifica ușor competența și autoritatea autorului conținutului ?
3. Se poate determina actualitatea conținutului (ultima actualizare și ce anume a fost actualizat) ?
4. Conținutul și locația sa sunt stabile (același URL) ?
5. Conținutul este credibil ?
6. Conținutul este consistent ?
7. Conținutul are valoare pentru comunitatea de utilizatori țintă ?
8. Este conținutul bine organizat ? Indexat ? Ușor de accesat ?

PROIECTAREA APLICAȚIILOR WEB – PRINCIPII OBIECTIVE PENTRU DESIGN

Simplitate și moderație

- Relativ la conținut, elemente vizuale, animație, organizare pagină, etc.

Consistență – pentru fiecare element al modelului proiectării.

- Reprezentări (stil text, grafică, culori, etc)
- Structuri hypermedia conform unor șabloane prestabilite
- Moduri de interacțiune, navigare și afișare conținut

Identitate

- Adaptarea elementelor de estetică și stil de navigare la domeniul aplicației și la categoria de utilizatori țintă.

Robustețe

- Conținut și funcționalitate relevante pentru utilizator.

Navigabilitate

- Navigare simplă și consistentă
- Navigare intuitivă și predictibilă

Atractivitate vizuală

- Aspect, look&feel pentru conținut, layout-ul interfeței
- Coordonarea culorilor, echilibrul între text, grafică și alte suporturi media
- Mecanisme de navigare

Compatibilitate

- Cu cât mai multe contexte (hardware, tipuri de conexiuni la Internet, SO, browser-e, etc.)

Principii specifice de proiectare a aplicațiilor web

- Procesarea cererilor aplicației
- Autentificare
- Autorizare
- Caching
- Gestiune excepții
- Jurnalizare și audit
- Navigare
- Page Layout
- Redare pagini
- Administrare sesiuni
- Validare informații

Principii specifice de proiectare a aplicațiilor web

Strategie ***procesare cereri*** : abordări

1. Comunicare prin forms

- Cod server-side ce generează cod HTML, starea view-ului asociat și logica de interacțiune cu browser-ul
- Potrivit pentru aplicații web bazate pe forms și care trebuie dezvoltate rapid (RAD)

2. Utilizare apeluri la servicii REST-ful

- Control mai fin asupra UI
- Mai multă flexibilitate în termeni de navigare, testabilitate (TDD), separare probleme
- Potrivit pentru aplicații ce necesită navigare flexibilă, tehnologii alternative de redare UI, în cazul folosirii TDD.

Principii specifice de proiectare a aplicațiilor web

Strategie ***procesare cereri*** – ghid

- Centralizare etape comune de pre și post procesare \Rightarrow reutilizare
- Separare logică de procesare de logică de prezentare : MVP, MVC
- Protejarea tuturor datelor sensibile trimise în rețele (atât externe cât și interne) : criptare, semnături digitale, SSL.

Șabloane de proiectare relevante:

- *Intercepting Filter* – lanț de filtre compuse pentru implementare procesări comune și acțiuni pre și post procesare.
- *Page Controller* – manipulare cerere pentru o anumită pagină sau acțiune.
- *Front Controller* – un singur obiect ce preia cererile și care poate fi modificat la runtime cu *decorators*.

Principii specifice de proiectare a aplicațiilor web

Autentificare

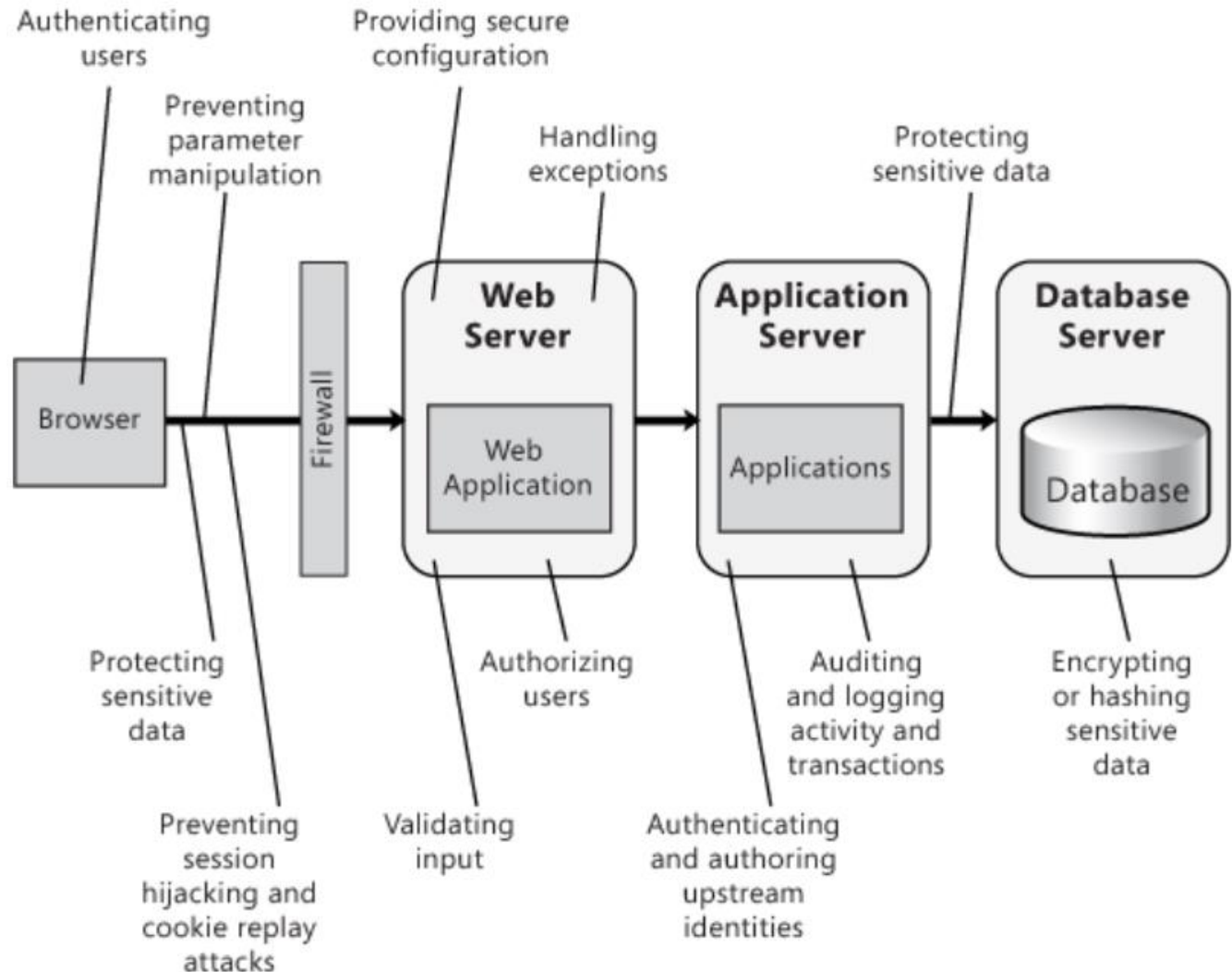
- Identificare zone de încredere în cadrul nivelelor aplicației ⇒ determinare necesități de autentificare utilizatori.
- Impunere de practici pentru securizare ca blocare cont, expirare parole, politici pentru parole puternice.
- Utilizare mecanism de autentificare oferit de platformă.
- Utilizare serviciu federat de autentificare sau SSO (single sign on) pentru a permite utilizarea aceluiași set de credențiale pe mai multe site-uri.
- Stocare parole codificate (hash, salted hash) în baza de date.
- Autentificare utilizatori de fiecare dată când se trece de la o zonă de încredere la alta.

Autorizare (activitățile accesibile unei entități autentificate)

- Autorizare la fiecare traversare de limite de încredere.
- Nivel potrivit de granularitate (overhead de administrare vs. flexibilitate redusă)
- Apărare în adâncime

Principii specifice de proiectare a aplicațiilor web

Probleme de securitate la o aplicație web tipică.



Principii specifice de proiectare a aplicațiilor web

Caching

- Încărcare date asincron sau folosind batch processing.
- Date în format utilizabil.
- Evitare date volatile și date senzitive necriptate.
- Pagini relativ statice sau părți relativ statice de pagini.
- Creare *rezervoare* (pools) de resurse costisitoare (ex. conexiuni de rețea), **nu** păstrarea lor în cache.

Șabloane de proiectare relevante:

- *Cache Dependency* – folosire informații externe pentru a determina starea datelor stocate în cache.
- *Page Cache* – avantaj la paginile dinamice accesate frecvent dar care se modifică rar și consumă resurse importante la construirea lor.

Principii specifice de proiectare a aplicațiilor web

Gestionare excepții

- Mesaje de eroare „user friendly”, eliminare date senzitive, adăugare informații suplimentare, curățare resurse și informații de stare.
- Nu capturați excepții pe care nu le tratați.
- Proiectare rutină globală pentru tratare excepții, ce afișează o pagină globală sau un mesaj de eroare pentru toate excepțiile netratate. Evitare folosire excepții personalizate (*custom*) dacă nu e necesar.
- Nu folosiți excepții pentru controlul fluxului aplicației.

Șablon de proiectare relevant:

- *Exception Shielding* – filtrarea datelor ce nu trebuie expuse.

Principii specifice de proiectare a aplicațiilor web

Jurnalizare (logging) și audit : în toate treptele aplicației.

Utilitate

- Indicație precoce a unui atac/amenințare de repudiere prin detectare activitate suspectă
- Răspuns la posibile solicitări în proceduri legale

Se urmăresc

- evenimente de administrare utilizatori
- evenimente critice de sistem
- operații business critice.
- activități neobișnuite

Restricționare acces la fișierele de *log* și *audit* (read-only pentru utilizatori)

Log și *audit* configurabile la instalare și în producție.

Nu se păstrează și nu se expun informații sensibile în fișierele de jurnalizare (log) și în fișierele de audit.

Șablon de proiectare relevant:

- *Provider* – API diferit de cel al clientului, pentru plug-in implementări personalizate.

Principii specifice de proiectare a aplicațiilor web

Navigare Separată de logica de procesare.

Structură de navigare consistentă (=> reduce complexitatea aparentă și confuzia utilizatorului)

Ghid:

- Încapsulare navigare într-o pagină master => consistentă de-a lungul aplicației.
- Asigurare navigare doar către "views" pentru care clientul este autorizat.
- Creare site-map.
- Utilizare elemente vizuale : embedded links, navigation menus, breadcrumb navigation (unde sunt, ce pot face, cum navighez rapid – site-uri mari cu ierarhii complexe de pagini).
- Utilizare "wizards" pentru implementare navigare între forms în mod predictibil.

Principii specifice de proiectare a aplicațiilor web

Organizare aspect pagini (Page Layout)

Separare *layout* de *componente* UI specifice și de *procesări* ale UI specifice.

Ghid:

- Utilizare CSS – în general; utilizare tabele (mai lent de redat, dar ca suport pentru grid-layout sau date reprezentate tabelar).
- Crearea și utilizarea unui *layout* comun.
- Creare și utilizare *master page* \Rightarrow aspect și comportament comun pentru toate paginile.
- Minimizare dimensiuni pagini (doar elementele specifice cererii curente).

Șabloane de proiectare relevante:

- *Composite View* - combinare views.
- *Template View*
- *Transform View* – transformarea datelor trimise nivelului Prezentare în cod HTML.
- *Two-Step View* – modelul datelor \rightarrow prezentare logică fără format specific \rightarrow formatul curent cerut.

Principii specifice de proiectare a aplicațiilor web

Redare pagini

Redare eficientă și maximizare utilizabilitate.

Ghid:

- Separare HTML, CSS, JS.
- Utilizare scripturi client-side.
- Utilizare AJAX (interactivitate sporită, procesare în background cu reducere număr de pagini reîncărcate).
- Utilizare opțiuni pentru *data binding* (ex. *DataSet*, obiecte personalizate).
- Utilizare *data-paging* pentru cantități mari de date.
- Suport pentru internaționalizare în componentele UI.
- Abstractizare tipuri componente GUI cu funcții de redare date și cu funcții de achiziții date.

Principii specifice de proiectare a aplicațiilor web

Administrare sesiuni

Ce, unde, cât timp stocăm ?

Ghid:

- Doar dacă și când e necesar.
- *Read-only* session.
- Alegere variantă adaptată la context :
 - în process (performanță, număr limitat de sesiuni concurente)
 - utilizare serviciu oferit de server Web (pentru date costisitor de recreat)
 - utilizare server SQL centralizat (în arhitecturi *Web farm*)

Protejarea (SSL, IPSec) canalului de comunicare cu serverul, separat pentru stocare sesiuni.

- Utilizare tipuri de bază pentru datele sesiunii pentru a reduce costurile de serializare.

Principii specifice de proiectare a aplicațiilor web

Validare informații

(probleme : SQL injection, stack overflow, etc.)

Ghid :

- Validarea tuturor datelor ce traversează zone de încredere.
- Asumare că toate datele controlate de client sunt rău intenționate și trebuie validate.
- Strategie de restricționare date rău intenționate.
- Validare dimensiune, domeniu de valori, format, tip.
- Validare și la client și la server
- Reutilizare reguli, software și șabloane de validare existente.

Evaluare formativă

1. Plecând de la informațiile sintetice de mai jos, realizați o analiză mai detaliată referitoare la păstrarea informațiilor sesiunilor în memoria volatilă a procesului server.

<https://forms.gle/yk2nZGXget6n2fEH9>

Alegere variantă de stocare date sesiuni adaptată la context :

- în process (performanță, număr limitat de sesiuni concurente)
- utilizare serviciu oferit de server Web (pentru date costisitor de recreat)
- utilizare server SQL centralizat (în arhitecturi *Web farm*)

OBS. Protejarea (SSL, IPSec) canalului de comunicare cu serverul utilizat separat pentru stocare sesiuni.

PLAN CURS

- Arhitectura și specificul sistemelor și aplicațiilor Web

Analiza aplicațiilor Web

- Modelul analiză pentru aplicații Web
- Analiza relații-navigare

Proiectarea aplicațiilor Web

- Principii
- Modelul proiect pentru aplicații Web

PROIECTAREA APLICAȚIILOR WEB – PRINCIPII

PIRAMIDA PROIECTĂRII APLICAȚIILOR WEB

Rezultatul proiectării – model ce conține elementele de **estetică**, **conținut** și **tehnologie**.

Proiectarea interfeței – structura și organizarea UI

- layout-ul ecranului
- modurile de interacțiune
- descrierea mecanismelor de navigare

Proiectarea esteticii (graficii) - LOOK&FEEL

- scheme de culori,
- geometria layout-ului,
- font, dimensiune și amplasare text,
- utilizarea graficii, etc.

Proiectarea conținutului:

structură
schiță conținut
obiectele conținut și relațiile dintre ele

Proiectarea arhitecturii:

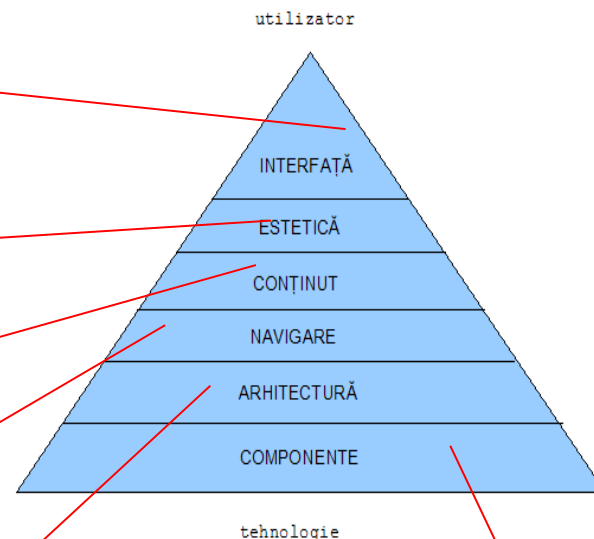
- Conținut - structura (hypermedia) de ansamblu a tuturor obiectelor conținut.
- Funcții - componente funcționale și relații.

Proiectarea navigării - fluxul navigării :

- între obiectele conținut
- pentru toate funcțiile aplicației

Proiectarea componentelor:

- Detalierea structurii interne și logicii de procesare necesară implementării componentelor funcționale.



CARACTERISTICI DEZIRABILE PENTRU INTERFAȚA UTILIZATOR

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

- simplu de utilizat
- ușor de învățat
- simplu de navigat
- intuitivă
- consistentă
- eficientă
- fără erori
- funcțională

PROIECTAREA INTERFEȚEI UTILIZATOR

La primul pas al proiectării se revizuiesc informațiile obținute în etapa de analiză:

- ierarhia utilizatorilor
- nevoile specifice fiecărei categorii de utilizatori, referitoare la
 - interacțiune,
 - funcționalitate
 - conținut.

În plus, utilizatorului trebuie să i se răspundă la 3 întrebări majore:

1. Unde mă aflu ?

Aplicația în cadrul căreia se află.
Poziția în cadrul ierarhiei conținutului.

2. Ce pot face acum ?

Funcții disponibile
Link-uri active
Conținut relevant

3. Unde am fost și unde mă duc ?

“Hartă” a navigării curente a utilizatorului.

PRINCIPII DE PROIECTARE A UI

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

“Bruce Tognozzi suggests...

Effective interfaces are *visually apparent* and *forgiving*, instilling in their users a *sense of control*. Users *quickly* see the breadth of their options, *grasp* how to achieve their goals, and do their work.

Effective interfaces *do not concern the user with the inner workings* of the system. Work is *carefully and continuously saved*, with full option for the user to *undo* any activity at any time.

Effective applications and services perform a *maximum of work*, while requiring a *minimum of information from users*.”

PRINCIPII DE PROIECTARE A UI

1. *Anticiparea* – anticiparea următoarei “mișcări” a utilizatorului și oferirea opțiunilor de navigare corespunzătoare.
2. *Comunicarea*
 - starea oricărei activități inițiată de utilizator,
 - identitatea utilizatorului,
 - poziția în ierarhia conținutului.
3. *Consistența* – relativ la reprezentare controale, menu, pictograme, estetică.
4. *Autonomia controlată* – facilitarea navigării, cu impunerea convențiilor de navigare stabilite pentru aplicație.
5. *Eficiența* – din perspectiva utilizatorului (nu a proiectantului).
6. *Flexibilitatea* – în raport cu diferite categorii de utilizatori; posibilitatea de a reveni din erori sau de pe căi de navigare alese greșit (UNDO).
7. *Focalizarea* – pe activitățile curente ale utilizatorului.
8. *Legea lui Fitt* – timpul de obținere a unei ținte este funcție de distanța la ea și de dimensiunea țintei \Rightarrow reducerea manevrelor de acces la obiecte de interfață ce trebuie acționate în secvență.
9. *Obiecte de interfață predefinite* – există o bibliotecă vastă.

PRINCIPII DE PROIECTARE A UI (cont.)

10. *Reducerea latenței* – multitasking; feedback audio la lansarea operațiilor la care răspunsul nu e imediat; afișarea unui control animat al progresului; eventual “entertainment”.
11. *Ușor de învățat* – design simplu și intuitiv; organizarea conținutului și funcționalității în categorii evidente pentru utilizator.
12. *Metafore* – utilizarea unei metafore corespunzătoare aplicației și utilizatorilor ei; imagini și concepte din experiența reală a utilizatorilor, augmentate cu facilități ajutătoare (ex. completare formular tip cu posibilități de selecție din liste).
13. *Păstrarea integrității* – salvare automată a tuturor datelor specificate de utilizator (ex. a formularelor completate) în vederea recuperării în cazul apariției unor erori.
14. *Lizibilitate* – font-uri, dimensiuni și culori bine alese.
15. *Urmărirea stării* – memorarea succesiunii stărilor interacțiunii cu utilizatorul în vederea continuării acesteia la o reconectare ulterioară.
16. *Navigare vizibilă* – crearea iluziei că aplicația vine în întâmpinarea utilizatorului, aducându-i la îndemână facilitățile de navigare necesare în contextul respectiv.

GHID PRACTIC DE PROIECTARE A UI

1. Viteza de citire de pe ecran este cu aprox. 25% mai lentă decât de pe hârtie \Rightarrow structurarea prezentării informațiilor text de dimensiuni mari.
2. Evitare prezentării paginilor “under construction”.
3. Prezentarea informațiilor cu evitarea nevoii de derulare.
4. Proiectarea consistentă a elementelor de navigare (menu, bare) și prezentarea acestora pe toate paginile.
5. Estetica trebuie folosită în sprijinirea funcționalității (nu invers).
6. Opțiunile de navigare trebuie să fie evidente chiar și pentru utilizatorii ocazionali.

Concluzie : interfață bine structurată și ergonomică.

MECANISME DE INTERFAȚĂ CU UTILIZATORUL

MECANISME DE INTERFAȚĂ PENTRU CONTROL

Obiective:

- “Fereastră” consistentă către conținut și funcționalitate
Proiectare estetică cu un look&feel coerent: layout și mecanisme de navigare.
- Ghidarea utilizatorului în cursul interacțiunii
Utilizarea unei metafore pentru înțelegerea intuitivă a interfeței.
- Organizarea opțiunilor de navigare și a conținutului.

MECANISME PENTRU IMPLEMENTAREA OPȚIUNILOR DE NAVIGARE:

- Menu-uri de navigare cu opțiuni organizate ierarhic.
- Pictograme – butoane, comutatoare, etc. pentru selectarea unei proprietăți sau a unei decizii.
- Imagini grafice – link-uri la obiecte conținut sau la funcționalitate.

FLUXUL OPERAȚIILOR DE PROIECTARE A UI

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

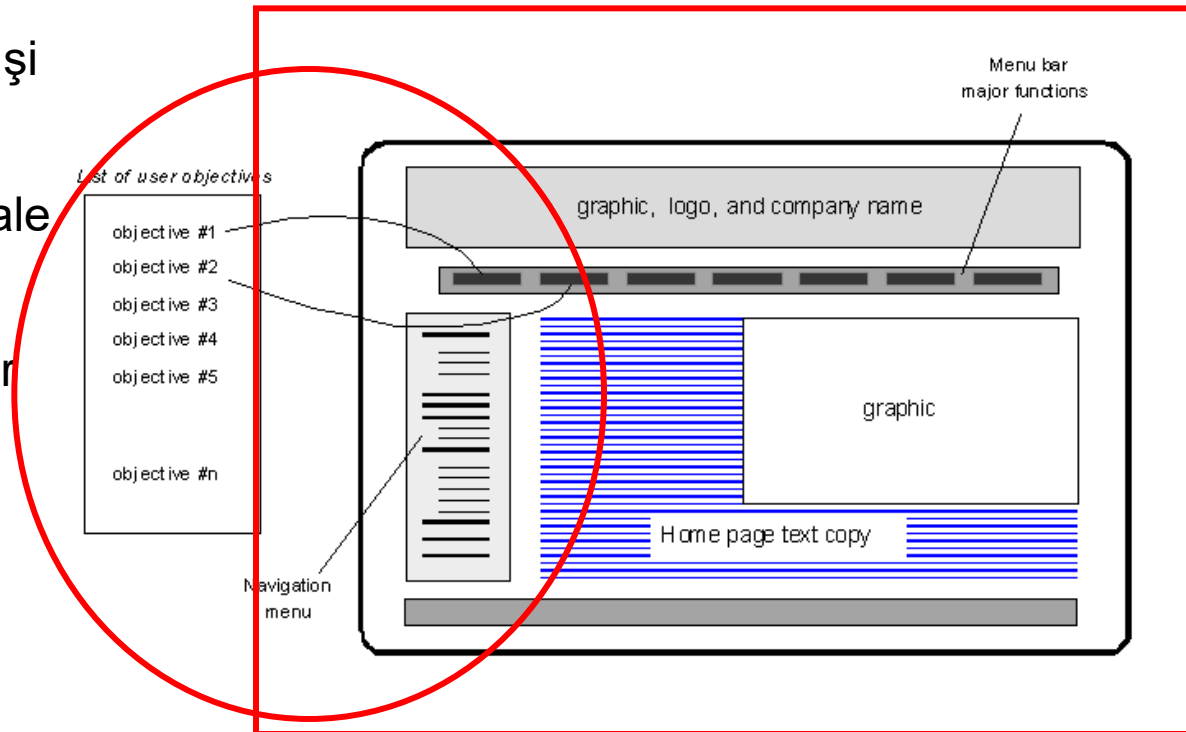
ANALIZA

Identificare cerințe utilizator, activități și context.
Crearea și analizarea cazurilor de utilizare pentru definirea
obiectelor conținut și a *funcționalității* (acțiuni).

1.Revizuirea modelului analiză și
rafinarea lui.

2.Realizarea unei schițe generale
a layout-ului interfeței.

3.Maparea obiectivelor utilizator
în acțiuni de interfață.



FLUXUL OPERAȚIILOR DE PROIECTARE A UI

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

4. Definirea unui set de *manevre utilizator* asociate cu fiecare acțiune de interfață; detalierea lor în *interacțiuni de interfață* (definite d.p.d.v. al *navigării*, *obiectelor conținut* și *funcțiilor aplicației*).
5. Creare a câte unui *set de imagini ale interfeței* pentru a ilustra *răspunsurile* la fiecare acțiune de interfață; prezentarea funcționalității și indicarea legăturilor de navigare.
6. *Rafinarea* interfeței și a seturilor de imagini cu elementele rezultate din proiectarea esteticii.
7. Identificarea *obiectelor utilizator* necesare în implementarea interfeței (clase specifice și clase de bibliotecă).

FLUXUL OPERAȚIILOR DE PROIECTARE A UI

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

8. Dezvoltarea unei *reprezentări procedurale* a interacțiunii utilizatorului cu interfața, utilizând diagrame de secvențe și de activitate.
9. Dezvoltarea unei *reprezentări comportamentale* a interfeței, cu diagrame de stări; definirea mecanismelor de control (obiecte și acțiuni) ce declanșează tranziții de stare.
10. Descrierea *layout-ului interfeței* pentru fiecare stare.
11. *Rafinarea și revizuirea* modelului proiect al interfeței, cu accent pe *utilizabilitate*.

PROIECTAREA ESTETICII

Layout

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

LAYOUT – ul

“Frumusețea este în ochii celui ce privește”.

PRINCIPII GENERALE:

- Evitarea supraîncărcării paginii
- Conținut 80% vs. Navigare și alte caracteristici 20%
- Organizare de la stânga sus la dreapta jos, conform priorității elementelor
- Grupare (conținut, navigare, funcții) conform unor șabloane recognoscibile de către utilizator.
- Evitarea necesității de derulare (scroll) pentru acces la totalitatea informațiilor unui obiect.
- Specificarea dimensiunilor elementelor layout-lui în valori relative la suprafața disponibilă.

PROIECTAREA ESTETICII

Look&Feel

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

DESIGN GRAFIC – LOOK&FEEL

“Frumusețea este în ochii celui ce privește”.

- aranjare în pagină (layout)
- scheme de culori
- fonturi, dimensiuni și stiluri
- utilizare de alte media (audio, video, animație)
- alte elemente de estetică.

PROIECTAREA CONȚINUTULUI

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

Două aspecte:

1. Proiectarea *obiectelor conținut* și a *mecanismelor necesare instanțierii relațiilor* dintre acestea – inginerul software.
2. Reprezentarea *informațiilor* în cadrul unui anume obiect conținut – proiectantul de conținut.

PROIECTAREA CONȚINUTULUI

OBIECTE CONȚINUT

Asemănătoare cu obiectele de date din software-ul convențional.

Atribute referitoare la:

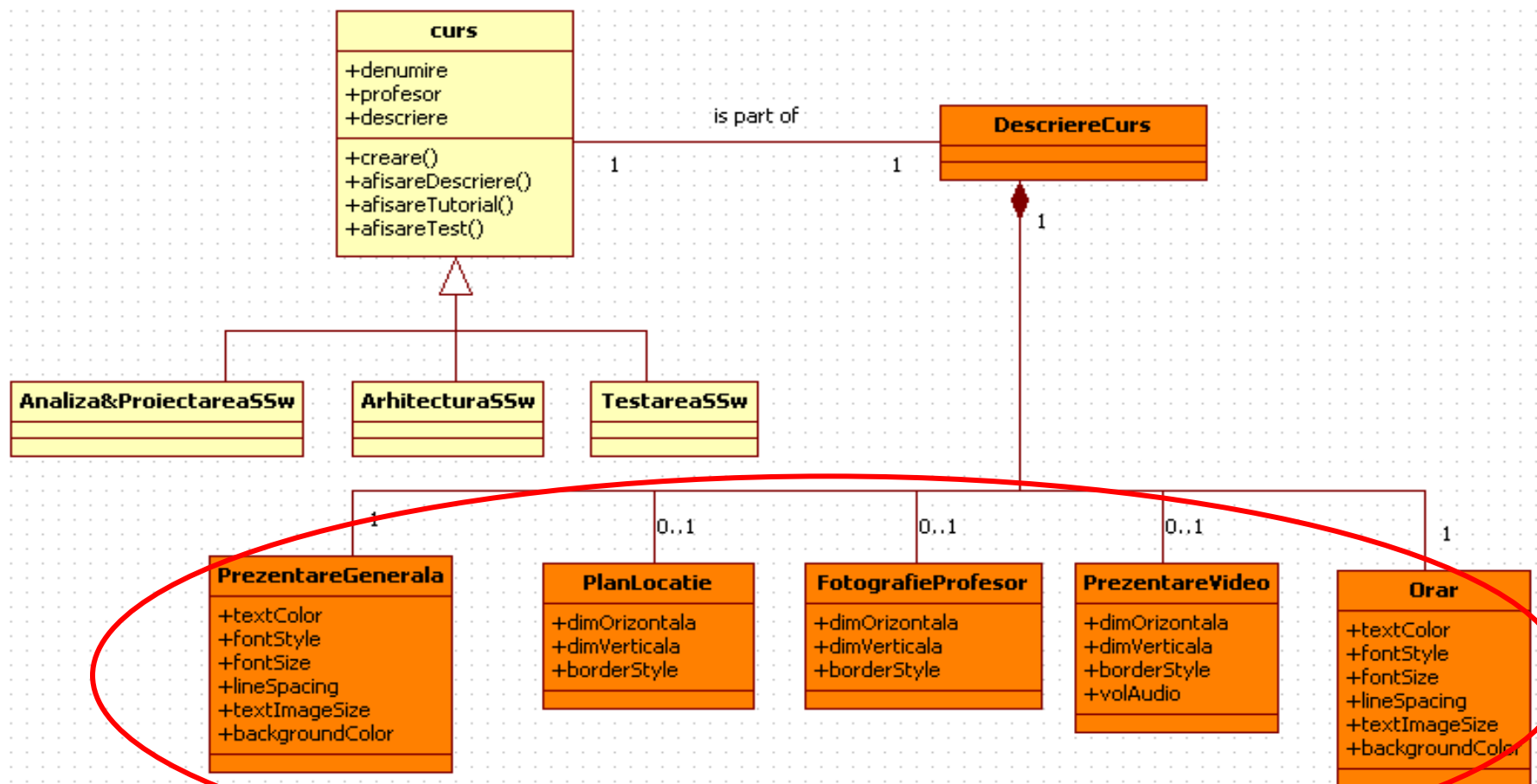
- Informația conținută (definite în modelul analiză)
- Implementare (definite la proiectare)

PROIECTARE CONȚINUT

- Schiță a informațiilor conținute
- Indicații de tip pentru obiectele conținut generice (ex. text descriptiv, imagini grafice, foto, etc.)
- Repartizarea obiectelor pe pagini
funcție de :
 - necesitățile de utilizare
 - restricții impuse de viteza de descărcare oferită de conexiunile la Internet
 - restricții impuse de volumul de derulare tolerat de utilizator.

PROIECTAREA CONȚINUTULUI

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor



OBIECTE CONȚINUT

PROIECTAREA ARHITECTURII

Proiectarea arhitecturii este legată de :

- Obiectivele stabilite pentru aplicație
- Utilizatori
- Filozofia de navigare stabilită

Arhitectura conținutului

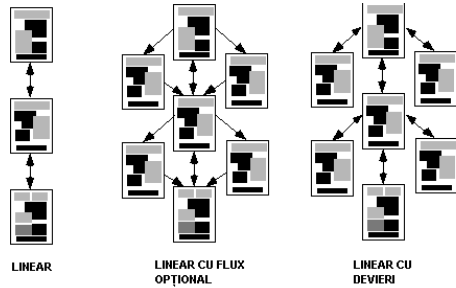
- Structurarea *obiectelor conținut* și a *obiectelor compuse* (ex. pagini Web) în vederea prezentării și navigării.
- Elemente de *etichetare*, *navigare* și *căutare*

Arhitectura aplicației

- Structurarea aplicației în vederea gestionării *interacțiunii cu utilizatorul*, manipulării *activităților interne* de procesare, efectuării *navigării* și *prezentării conținutului*.
- În paralel cu proiectarea UI, esteticii și a conținutului.

PROIECTAREA ARHITECTURII CONȚINUTULUI

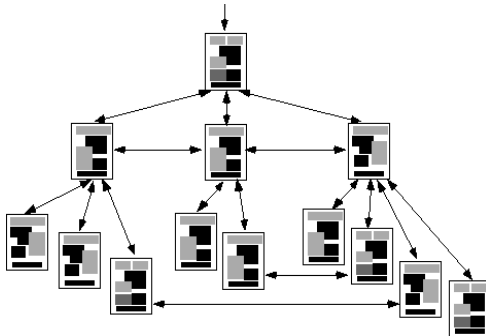
Tipuri de structuri



LINEARĂ

Dacă secvența interacțiunilor este predictibilă.

Variații: conținut alternativ, conținut complementar.



IERARHICĂ

Fluxul de control este dirijat de-a lungul ierarhiei, de sus în jos, precum și pe orizontală.

Proiectarea interfeței utilizator

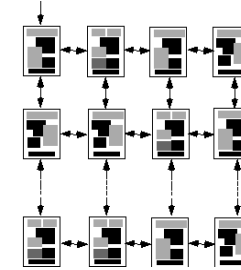
Proiectarea esteticii

Proiectarea conținutului

Proiectarea arhitecturii

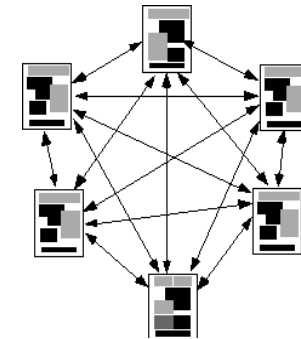
Proiectarea navigării

Proiectarea componentelor



GRID

Organizare conținut pe categorii în 2 sau mai multe dimensiuni.



WEB generală

Comunicare posibilă între orice 2 pagini, utilizând hyperlink.
Permite flexibilitate maximă în navigare.
Poate induce confuzii.

PROIECTAREA ARHITECTURII APLICAȚIEI

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

ARHITECTURA CONȚINUTULUI

Pentru *sistemele complexe* se utilizează structuri compuse din *combinații ale tipurilor* prezentate anterior.

Ex. Arhitectura de ansamblu poate fi ierarhică, cu subelemente având caracteristici ale altor tipuri de structuri.

Obiectivul proiectantului arhitecturii: realizarea unei arhitecturi potrivită cu conținutul de prezentat și cu procesarea cerută.

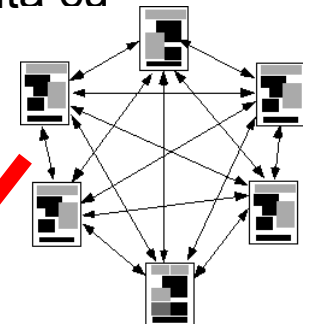
ARHITECTURĂ DE APLICAȚIE WEB

Arhitectură pe 3 nivele cu separare:

- Interfață utilizator (view)
- Funcționalitate (controller)
- Conținut informațional (model)

Avantaje:

- Simplifică implementarea
- Extinde reutilizarea



ARHITECTURA
Model View Controller (MVC)

PROIECTAREA ARHITECTURII APLICAȚIEI

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

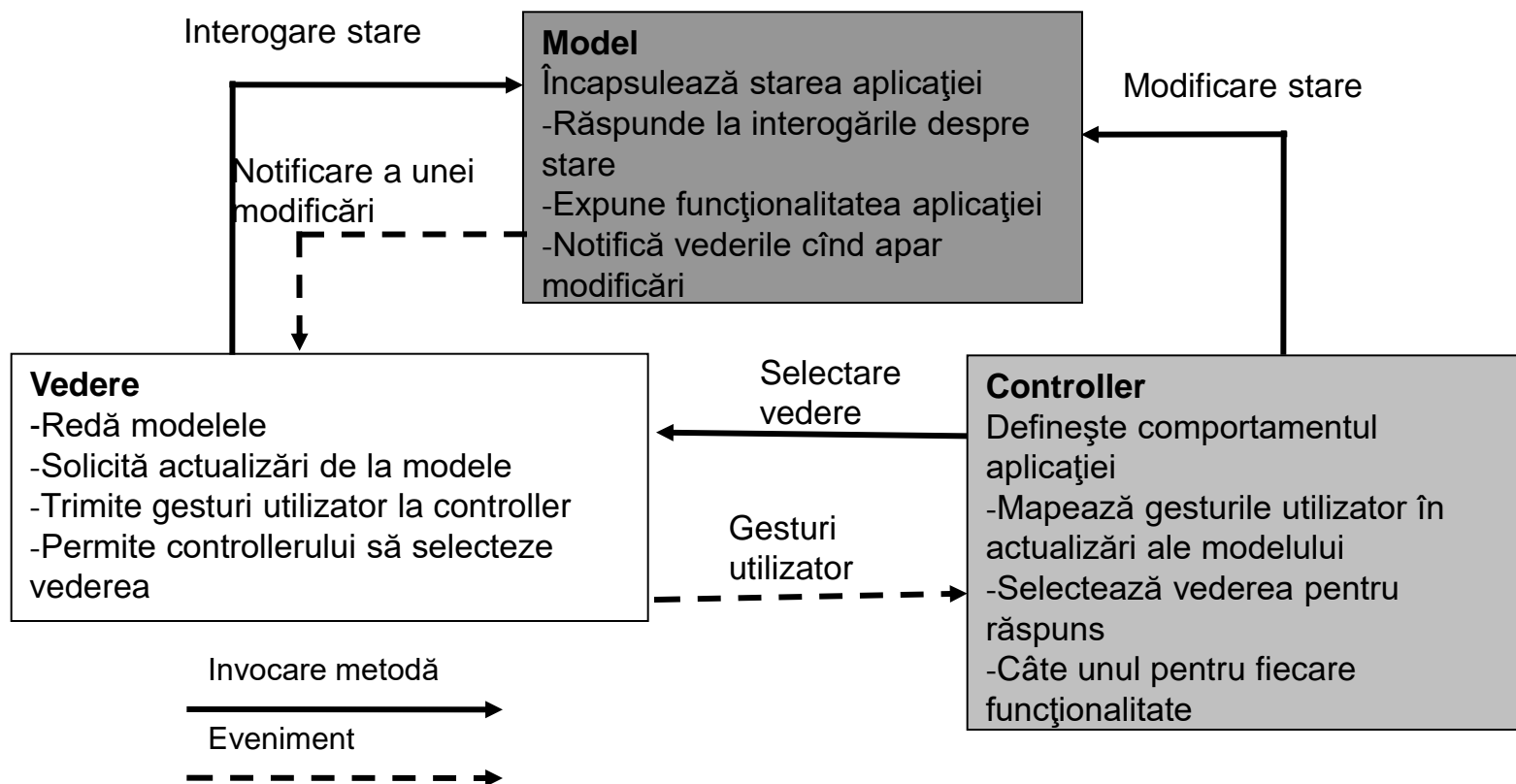
ARHITECTURA MVC

- *model* - conținutul și logica de procesare specifice aplicației
 - obiectele conținut
 - accesul la surse externe de date și informații
 - funcționalitatea de procesare specifică aplicației
- *view* – conține funcțiile de interfață și permite
 - prezentarea conținutului și a logicii de procesare
 - accesul la funcționalitatea de procesare solicitată de utilizatorul final.
- *controller*
 - gestionează accesul la model și la view
 - coordonează fluxul de date dintre acestea.

PROIECTAREA ARHITECTURII APLICAȚIEI

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

ARHITECTURA MVC



View este actualizat de către *controller* cu date din *model* pe baza intrărilor utilizatorului.

PROIECTAREA NAVIGĂRII

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

Precedată de:

- Stabilirea arhitecturii aplicației Web
- Identificarea componentelor arhitecturii (pagini, scripturi, applet-uri, alte funcții de procesare)

Obiectiv :

- Definirea *căilor de navigare* ce permit accesul la *conținut* și la *funcționalitate*.

Metodă:

- Identificarea *semanticilor de navigare* pentru diferiți utilizatori
- Definirea *mecanismelor (sintactica) de navigare*

PROIECTAREA NAVIGĂRII

SEMANTICA NAVIGĂRII

- Începe cu considerarea *ierarhiei utilizatorilor* și a *cazurilor de utilizare* corespunzătoare.

Fiecare actor poate folosi aplicația Web în mod diferit, având deci necesități diferite de navigare.

- În cursul interacțiunii cu aplicația Web se întâlnesc o serie de *unități semantice de navigare* (NSU).

Def. NSU = set de informații și structurile de navigare care colaborează pentru îndeplinirea unui subset de cerințe utilizator corelate.

Concluzie : Navigare orientată pe activitățile utilizatorului, nu doar pentru căutare și găsim de informații.

Detalii utile la :

<https://www.slideshare.net/aimeemaree/cognitive-analysis-of-web-site-navigation>

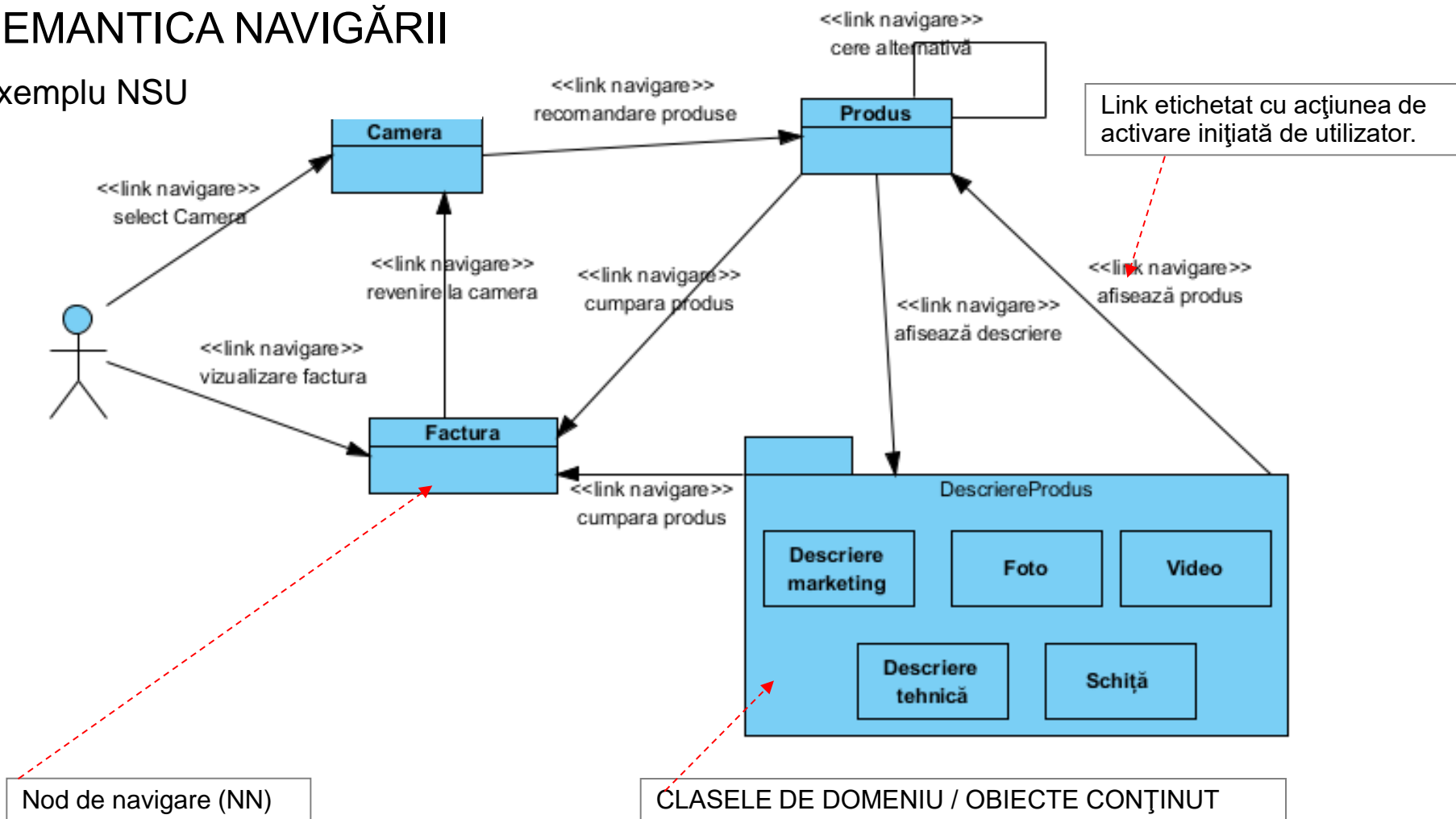
<https://www.smashingmagazine.com/2011/06/planning-and-implementing-website-navigation/>

PROIECTAREA NAVIGĂRII

- Proiectarea interfeței utilizator
- Proiectarea esteticii
- Proiectarea conținutului
- Proiectarea arhitecturii
- Proiectarea navigării
- Proiectarea componentelor

SEMANTICA NAVIGĂRII

Exemplu NSU



PROIECTAREA NAVIGĂRII

SINTAXA NAVIGĂRII

MECANISME DE NAVIGARE (exemple semnificative)

- Legătură individuală de navigare (text, pictogramă, buton, metaforă vizuală)
- Bară de navigare orizontală – categoriile majore de conținut sau funcționalitate (în general, 4-7 categorii)
- Coloană verticală de navigare – (1) categorii majore de conținut sau afișare; (2) obiectele conținut majore (expandare ierarhie)
- Tab – categorii de conținut sau funcționalitate
- Site map – cuprins al navigării către toate obiectele conținut și funcțiile aplicației.

CONVENȚII DE REPREZENTARE A SINTAXEI DE NAVIGARE (exemple):

- Butoane în reprezentare tridimensională (clickable)
- Feedback audio sau vizual
- Culori pentru link-urile reprezentate textual

Exemple de bune practici de proiectare : <https://www.crazyegg.com/blog/website-navigation/>

PROIECTAREA COMPONENTELOR

Proiectarea interfeței utilizator
Proiectarea esteticii
Proiectarea conținutului
Proiectarea arhitecturii
Proiectarea navigării
Proiectarea componentelor

PROIECTAREA COMPONENTELOR

Proiectare similară cu cea a componentelor pentru software convențional.

Funcționalități tipice ale componentelor aplicațiilor Web:

- Procesare pentru generarea dinamică de conținut.
- Procesare pentru generarea dinamică de capacități de navigare.
- Calcule și procesare date corespunzătoare domeniului aplicației.
- Acces și interogări sofisticate ale bazelor de date.
- Stabilire de interfețe de date cu sisteme externe.

PLATFORMA DE LIVRARE

Opțiuni pentru platforma de livrare:

- bazată web,
- mobile,
- ambele

Probleme și soluții pentru varianta mobile:

Conectivitate intermitentă \Rightarrow oferire de serviciu limitat fără conectivitate la rețea

Putere procesor \Rightarrow minimizare operații computațional-intensive.

Management alimentare \Rightarrow minimizare energie folosită de aplicație.

Tastatură on-screen \Rightarrow minimizare interacțiune prin tastatură.

Bună practică – separare versiuni *browser-based* și *mobile*, ale front-end.

Obs. Este posibil să fie necesare descompuneri arhitecturale complet diferite pentru aceste versiuni diferite astfel încât să se asigure păstrarea performanței și a altor caracteristici.

INSTRUMENTE DE DEZVOLTARE cu REUTILIZARE

Categorii de tehnologii de dezvoltare cu reutilizare:

- mobile development toolkit
- web application framework

Dezvoltatorul trebuie să se conformeze premizelor *built-in* referitoare la arhitecturile aplicațiilor software.

Tehnologia de dezvoltare poate avea o influență indirectă asupra arhitecturii sistemului.

Obs. Dezvoltatorii favorizează de obicei alegerile arhitecturale care folosesc tehnologii ce le sunt familiare. De exemplu, dacă echipa are multă experiență cu baze de date relaționale, ar putea să insiste pe folosirea acestui tip în locul unei baze de date NoSQL.

Evaluare formativă

1. Enumerați elementele modelului rezultat al proiectării aplicațiilor web.
2. Enumerați câteva principii de proiectare a interfeței cu utilizatorul în aplicațiile web.
3. Ce proiectează inginerul software referitor la conținutul aplicațiilor web ?

<https://forms.gle/GsGh8biRtGM82FCB8>

REZUMAT

Specificul sistemelor și aplicațiilor Web

- Utilizarea intensivă a rețelelor, concurență utilizatori, încărcare nepredictibilă,
- Determinate de date, conținut dinamic, lansare rapidă pe piață, evoluție continuă
- Cerințe de calitate: utilizabilitate, performanță, disponibilitate, securitate
- Categori: informaționale, sociale, orientate pe servicii, portal, acces la baze de date (tranzacții)
- Procesul WebE: creare model analiză și model proiect.

Recomandări practice pentru dezvoltarea aplicațiilor Web

- Domeniul aplicației, scenarii, plan proiect, revizuri, instrumente și tehnologii adecvate, teste exhaustive

REZUMAT

Analiza aplicațiilor Web

Analiza cerințelor aplicațiilor Web

- Identificare categorii de utilizatori, cerințe utilizator, context și interacțiuni.
- UCs => *obiectele conținut* (entități) și *funcționalitățile* (acțiuni).

Modelul analiză pentru aplicații Web

- **Conținut:** obiecte de tip conținut, relații, ierarhii
- **Funcții:** observabile de către utilizator, interne (implementare funcții observabile)
- **Interacțiuni:** UCs, diagrame de secvențe, diagrame de stări, prototipul interfeței utilizator
- **Configurație:** infrastructură, instalare, interoperabilitate

Analiza relații-navigare

- Rafinarea relațiilor între elemente: ierarhie stakeholderi, relații între obiecte conținut și elemente funcționale, navigare la elemente sau grupuri de elemente, evaluare

REZUMAT

Proiectarea aplicațiilor Web

Principii

- Activități : tehnice (ingineri software), non-tehnice (designeri de grafică, dezvoltatori de conținut)
- Calități: utilizabilitate, fiabilitate, eficiență, mentenabilitate, securitate, disponibilitate, scalabilitate, timp scurt lansare pe piață
- Obiective: simplitate și moderație, consistență stil (reprezentare, interacțiune, navigare), identitate, robustețe, navigabilitate, atractivitate vizuală, compatibilitate cu agenții client

Modelul proiect pentru aplicații Web

Proiectare: interacțiune, funcționalitate, conținut

- **UI:** cerințe, principii, fluxul operațiilor de proiectare
- **Estetica:** layout, look&feel
- **Conținutul:** obiecte conținut și mecanisme instanțiere relații între acestea, informații și tipurile lor
- **Arhitectura:** conținut (lineară, grid, ierarhică, web), aplicație (MVC)
- **Navigarea:** unități semantice de navigare corelate cu UCs, sintaxa (mecanisme și convenții de reprezentare)
- **Componentele:** funcționalități tipice