** Ministerul Educaţiei Republicii Moldovei**

**Universitatea Tehnică a Moldovei**

[**Departamentul**](https://utm.md/subdiviziuni-universitare/facultati/facultatea-calculatoare-informatica-si-microelectronica/catedra-calculatoare/) **informatica si ingineria sistemelor**

**Raport**

**La proiectul de practica**

A elaborat: Nicolenco Eugeniu, Gr. C-162

Mentor: Alexei Corduneanu, SRL „Vivat Consulting”

2018

Introducere

Marele tendințe ce apar asăzi în lumea IT creează oportunități pentru studenți și cerințe pe piață așa cum ar fi elaborarea aplicațiilor web responsive și flexibile.

Elaborarea web API presupune parcurgerea a mai multor pași. Exista căi ușoare, scurte, există căi mai lungi dar sigure. Prin cale sigură se înțelege elaborarea aplicației flexibile, ușor de extins și de manipulat cu conținutul ei. Calea ușoară presupune elaborarea aplicației direct în fișierele de bază, de execuție, cu un număr fix de funcții și conținut, astfel de aplicații sunt greu de extins sau de corectat. Pașii de elaborare a aplicației flexibile sunt deduși în mersul lucrarii și exemplificați prin obiect sau explicație de conținut rezonabil.

Ca obiect al lucrării practice am ales de a elabora o aplicație web ce ar prezenta fața unei companii ce prestează servicii foto – video, care ar cuprinde posibilități de contact, promovare, informare și comandă a serviciilor presupuse. Aplicația va fi stilată conform tendințelor și cerințelor contemporane.

Lucrarea este împărțită pe capitole și teme așa cum am lucrat și am învățat în timp. Fiecare capitol este destinat unei părți al dezvoltării aplicației precum și al dezvoltării duhui de echipă ceea ce este nu mai puțin de important pentru a dezvolta o aplicație în termen și cu o funcționalitate modernă; schimbul de idei între colegi, co-dezvoltatori este opțiunea care ne ajută să găsim soluții mai bune chiar atunci cînd ne pare că sunt ascunse de noi și altă cale nu există.

Pentru lucru în echipă este necesar de a stabili niște standarte de comunicare, de programare, care convin tuturor și nu duce colectivul în eroare; pentru astfel de sarcini lumea IT deja a propus o mulțime de soluții și standarte pentru dezvoltatori, care sunt valabile pentru orice limbaj sau mediu de programare; Colecția de instrumente și reguli vor fi descrise în conținutul lucrării și descrise adițional.

**Capitolul I**

* 1. Comunicarea corporativă

Comunicarea este partea fundamentală a relațiilor inter-personale ce compune orice societate, comunitate, grup. Comunicarea realizează legături între diferite părți ale sistemelor precum și diferite sisteme.

Comunicarea într-o corporație presupune conducerea/subordonarea, managementul, etc. Astfel există și reguli de comunicare într-o întreprindere și relații între colegi, relații cu parteneri. Comunicarea înseamnă aproape totul în management, atîta timp cît de calitatea comunicării depinde înțelegerea problemelor cu care se confruntă orice angajat.

Comunicarea verbală cît și convobirea prin telefon ține de cîteva da și nu ce trebuie respectate:

Da –

* Convorbirea trebuie să fie fluentă, clară și destul de sonor într-un limbaj cunoscut de toți participanții.
* La comunicarea în grup trebuie de adresat cui ne adresăm, chiar dacă vorbim întregului grup, e bine să spunem ”Dragi prieteni” sau ”stimate public”.
* Cînd începem convorbirea cu cineva stăin e bine să ne prezentăm primul.

Nu –

* Nu mestecăm în timul convorbirii.
* Nu oferim prea multă informație în perioade scurte de timp.
* Nu folosim abreviaturi în convorbire.

și multe altele.

1.1.2 Comunicarea între corporative/filiale

Cea mai extinsă metodă de comunicare între parteneri/colegi este expedierea e-mail-urilor sau cum se mai numește poșta electronica, care se foloseste pentru convorbiri unu la unu, unu la mai mulți sau conferență.

Comunicarea prin mail are foarte bune oportunități comparativ comunicării verbale sau telefonice, ce ține de securitate, necesitate, conținut, acoperire și altele.

Securitatea convorbirii este ceea pentru ce opteză toți utilizatorii web, și este acel factor de ce folosesc poșta electronică. Securiatatea mail-urilor poate fi aranjată de criptare, expedierea prin canaluri private, VPN etc, tot aici mai putem spune că mail-urile pot fi securizate prin nivele de acces sau parolă.

Necesitatea comunicării prin mail apare atunci cînd avem de expediat fișiere, mesaje cu conținut masiv. Este o bună alegere pentru a expedia sarcini subordonaților sau formulare de completat la grupuri de utilizatori. La fel putem spune că necesitatea putem a o numi ”urgență”, deci putem lăsa un mesaj pentru a fi citit și răspuns mai tîrziu.

Conținutul mesajului poate fi diferit, aceasta este o opțiune foarte bine utilizată de administratori ale diferitor servicii care promovează mesaje cu conținut similar paginilor web, tabele excel, dări de seamă și multe alte feluri de vizualizări.

Acoperirea este elementul nedesponibil chiar și multor messengerelor contemporane, mesajele electronice pot fi accesate din orice loc al lumii unde există internet.

Cele mai importante dialoguri fie între diferite corporații fie între colegi, deoarece mesajele se salvează în diferite servicii, fie depozitele deținătorilor serviciilor mail (*Gmail*) sau depozite locale, fiecare utilizator deține istoria proprie a mesajelor și istoria înmărțită de alți utilizatori, și pot fi accesate în orice timp de oriunde, și pot fi folosite pentru amintire a evenimentelor din trecut sau ca indicii pentru pașii următori.

* 1. Programarea în echipă

Programarea în echipă este modalitatea de funcționare a tutror întreprinderilor în domeniul IT, deoarece lucrînd la un proiect mare este imposibil totul să fie făcut de un singur om. În prezent echipele se structurează pe departamente și subdepartamente în care fiecare are funcțiunea lui; pentru funcționarea corectă și efectivă a unei echipe este necesar ca fiecare participant să introduce un impact în lucru, o idee, o soluție etc.

Pentru un lucru efectiv și economie de timp la programatori se stabilesc cîteva standarte fie pnetru un timp îndelungat în companie, fie doar pe proectul curent. Echipa stabilește șablonul de programare, după ce se propun soluții de dezvoltare și menținere a proiectului din care se aleg cele mai convenabile, cele mai performante. Se stabilește forma creerii variabilelor, funcțiilor, componentelor și structurilor de date, această opțiune este numită *naming conventions* – un set întreg de convenții care se stabilesc fie de echipă/șef de echipă sau de client/reprezentant al comenzii. Pot fi multe tipuri de convenții, însă există seturi cunoscute global care vor fi descrise mai jos:

* camelCaseConventions / \_snake-conventions / slash-conventions – tipuri de variabile (mereu prima literă este mică ”a” așa tip se numește lowerCase – specific varibilelor)
* CamelCaseConvention / … - tipuri de clase în limbaje orientate pe obiecte (prima literă este mare ”A” – specific claselor)

Deasemenea pot fi folosite și diferite prefixe pentru amintirea tipului variabilei, clasei sau metodei. Spre exemplu limbajul PHP strict utilizează simbolul $ pentru orice tip declarabil de programist.

* CAPS\_CASE – se folosește cam rar dar este preferată de unele limbaje de programare cum ar fi Rust, Python, Ruby pentru a specifica date constante.

1.2.2 Șabloane de programare

În programare există multe tipuri de șabloane, primele au fost introduse chiar la începutul erei de programare, acestea au fost – programarea declarativă, logică și programarea orientată pe obiecte.

Acum piața programării este acoperită în majoritar de programarea orientată pe obiecte, însă acum programarea orientată pe obiecte a devenit un domeniu sau o categorie ce cuprinde varietăți de șabloane de priectare și design pentru a face programarea mai efectivă și mai puțin costisitoare în timp.

Șablon de design – așa mai numit *design pattern* este abordare a soluțiilor programării care cu timpul a devenit ceva necesar la crearea unei API de orice complexitate. Alegerea pattern-ului de proiectarea a sistemei îi poate veni fie liderului echipei fie reprezentatului comenzii.

Design pattern-urile acum se utilizează ca elemente de bază, asemenea unor metode incluse în limbaje bine dezvoltate, în framework-uri dezvoltate pe diferite domenii, fiew web sau desktop, în care se include:

* creator de obiecte (prin ierarhii de obiecte) care au incluse metode de bază și pot defini metode noi.
* obiect fără pereche (singleton) – obiecte unice într-o aplicație (cum ar fi ”Cursor”)
* compoziție de obiecte
* decorator și altele

De bază design pattern-urile se împart pe 3 mari categorii: structurale, de creare și de corecție(sau de comportament) din care cele mai semnificative și des utilizate (pe lîngă cele exemplificate mai sus) putem marca următoarele: Iterator, observer, statut, adapter, proxy,punte, prototype, abstract factory (crearea obiectelor abstracte).

Șabloanele de arhitectură însă își extind infulența asupra dezvoltării aplicației decît pe modul de abordare a soluțiilor, și conține idea de structurare a codului, fișierelor și bibliotecilor de date separat.

MVC – (Model-view-controller) – este un pattern des aplicat în construcția aplicațiilor responsive în care se separă model-ul (date, informații din baze sau fișiere) de view (suprafața vizualizată de utilizator [User interface]) și de controller (unitatea ce controlează acțiunile utilizatorului și produce răspuns) una de alta în fișiere aparte în așa fel că modificarea fiecărui component se realizează independent de celelalte. Scopul de bază a acestui concept de proiectare constă în separarea *business*-logicii aplicației de vizualizarea utilizarea acestui șablon permite reutilizarea codului, extinderea lui, la fel cum și întreaga aplicație. Astfe Cel mai semnificativ plus a acestui gen de programare este că la dezvoltarea fiecărui component aparte poate lucra un om sau o echipă, după structura și convențiile stabilite. Astfel de arhitectură face dezvoltarea, ba chiar și aplicația destul de flexibilă și reduce numărul de pietre ce se pot întîlni pe calea realizării aplicației. Deoarece MVC nu are strictețe în utilizare, poate fi utilizat în mod diferit, nu exită un decensămînt stabilit unde trebuie să fie aplicată *business-*logica, ea poate fi aplicată și în controller și în model. MVC utilizează destul de multe șabloane de design, din care cele necesare sunt: Compozitorul (crează o ierarhie din obiecte în formă arborescentă), Observatorul (pentru utilizarea acțiunilor utilizatorului) și Strategie(propune încapsularea datelor, ce și crează separarea componentelor una de alta).

MVC – este alegerea bună pentru începători pentru că conține ajutor ascuns în comportamentul său prin delegarea fiecărui component anumite părți a funcționării; Astăzi există multe instrumente bazate pe MVC care deasemenea sînt ușor de implementat și utilizat, o oportunitate indispensabilă pentru începători în porgramare și amatori. *Schema MVC în anexa 1*

MVVM – (Model-View-ViewModel) Asemănător MVC, a fost proiectat ca moștenire de la MVP (Model-View-Presenter) este orientat pe platforme de dezvoltare așa cum Windows Presentation Foundation, Silverlight și ZK framework. La fel se utilizează pentru separarea componentelor una de alta, spre deosebire de alte pattern-uri în MVVM apare ViewModel care cel mai des (dar nu obligatoriu) conține *business-*logica unde se conține modul de funcționarea a aplicației și comportamentul în conformitate cu acțiunile utilizatorului. ViewModeul permite aplicarea conceptului de legare a datelor în două direcții, spre exemplu completarea formei de logare îndată produce răspunsuri pentru corectitudinea completării și reacționează la fiecare schimbare aplicată de către utilizator.

La folosirea acestui pattern view (componenta de vizualizare) nu produce o interfață visibilă direct dar transferă această obligație ViewModel-ului care prin legarea fluxului de date în ambele sensuri produce această interfață pentru a fi mai departă afișată conform parametrilor setați în view.

*Schema MVVM în Anexa 2.*

PAC - (Presentation-abstraction-control) – asemănător MVC are 3 componente de bază, presentation – asemenea View – relizează interfața utilizatorului, abstraction – asemenea model – colecționează și procesează date, și control (asemenea controller) care produce *bussiness-*logica aplicației. Spre diferența de MVC, PAC este utilizată ca o structură ierarhicală de agenți, fiecare consistînd din încă o triadă din componente (PAC). Agenții comunică între ei prin partea de control a fiecărui participant, are ca un mare plus – încărcarea foarte rapidă a aplicației și afișarea ageră a interfeței utilizatorului.

*Schema PAC în Anexa 3.*

HMVC – (Hierarchical-Model-View-Controller) – asemenea PAC doar că conține agenți compuși din MVC.

Indiferent de șablonul de arhitectură ales, ehipa trebuie să fie cunoscută cu fiecare din ele pentru că este posbil comanda să fie pe un anumit șablon, și ar fi inadmisibil ca echipa să nu cunoască soluțiile și modul de utilizare a unui-a din ele. Uneori e destul cunoașterea unui din ele și astfel e foarte apropiată și structura celorlalte pe motivul asemănării lor, și componentelor chiar complet comune.

N Layered Architecture – se mai numește N tiers arhictecture.