

Universitatea *Transilvania* din Bras,ov

Facultatea de Matematică s,i Informatică

Programul de studii: Informatică aplicată

Lucrare de Licent,ă

Autor: Vlad Bianca - Lucia

Coordonator s,tiint,ific: Lect. Univ. Dr. Bocu Răzvan

BRAS,OV

2015

2



Universitatea *Transilvania* din Bras,ov Facultatea de Matematică s,i Informatică Programul de licent,ă: Informatică aplicată

PromON

2

Cuprins

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Introducere | | 7 |
|  | 1.1 | Scurtă prezentare a aplicat,iei . . . . . . . . . . . . . . . . . . | 7 |
|  | 1.2 | De ce Android? . . . . . . . . . . . . . . . . . . . . . . . . . . | 8 |
| 2 | Platforma Android | | 11 |
|  | 2.1 | Istoric . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 11 |

1. Arhitectura platformei . . . . . . . . . . . . . . . . . . . . . . 11

2.3 Linux Kernel . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

1. Native Libraries . . . . . . . . . . . . . . . . . . . . . . . . . . 15

2.5 Android Runtimes . . . . . . . . . . . . . . . . . . . . . . . . 15

1. Android Applications . . . . . . . . . . . . . . . . . . . . . . . 16
2. Versiuni Android . . . . . . . . . . . . . . . . . . . . . . . . . 17
3. Android Platform Fragmentation . . . . . . . . . . . . . . . . 19

|  |  |
| --- | --- |
| 2.9 Android Support Library . . . . . . . . . . . . . . . . . . . . | 19 |
| 3 Componetele unei aplicaţii Android | 21 |

1. Activity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
2. Crearea unei activităţi . . . . . . . . . . . . . . . . . . 22

3.1.2 Declararea unei activităţi . . . . . . . . . . . . . . . . 22

1. Ciclul de viaţă al unei activităţi . . . . . . . . . . . . . 23

3.2 Intent . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23

1. Service . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
2. Crearea unui Service . . . . . . . . . . . . . . . . . . . 24

3.3.2 Declararea unui Service . . . . . . . . . . . . . . . . . 24

* 1. Ciclul de viaţă al unui Service . . . . . . . . . . . . . . 24

1. Content Provider . . . . . . . . . . . . . . . . . . . . . . . . . 25
   1. Crearea unui Content Provider . . . . . . . . . . . . . 25

|  |  |
| --- | --- |
| 3.5 Broadcast Messages . . . . . . . . . . . . . . . . . . . . . . . | 26 |
| 4 Limbajul de programare Java | 27 |

1. Platforma Java . . . . . . . . . . . . . . . . . . . . . . . . . . 27
2. Configurarea unui computer . . . . . . . . . . . . . . . . . . . 28
3. Limbaj de programare obiect orientat . . . . . . . . . . . . . . 29

3

4 *CUPRINS*

1. Abstractizarea datelor . . . . . . . . . . . . . . . . . . 29
2. Încapsularea . . . . . . . . . . . . . . . . . . . . . . . . 30

4.3.3 Mos,tenirea . . . . . . . . . . . . . . . . . . . . . . . . 30

1. Suprascrierea unei metode . . . . . . . . . . . . . . . . 31

4.3.5 Adnotarea @Overriding . . . . . . . . . . . . . . . . . 32

1. Polimorfismul . . . . . . . . . . . . . . . . . . . . . . . 32
2. Clasa de bază universală . . . . . . . . . . . . . . . . . 33

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | JSOUP API | | | 37 |
| 6 | ASP.NET Web API | | | 41 |
|  | 6.1 | REST . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 41 |
|  | 6.2 | Uniform Interface . . . . . . . . . . . . . . . . . . . . . . . . . | | 41 |
|  | 6.3 | Stateless . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 42 |
|  | 6.4 | Cache-able . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 42 |
|  | 6.5 | Client-server . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 42 |
|  | 6.6 | Layered System . . . . . . . . . . . . . . . . . . . . . . . . . . | | 42 |
|  | 6.7 | Code on Demand . . . . . . . . . . . . . . . . . . . . . . . . . | | 43 |
|  |  | 6.7.1 GET . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 44 |
|  |  | 6.7.2 | POST . . . . . . . . . . . . . . . . . . . . . . . . . . . | 45 |
|  |  | 6.7.3 PUT . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 45 |
|  |  | 6.7.4 | DELETE . . . . . . . . . . . . . . . . . . . . . . . . . | 46 |
|  | 6.8 | Proiect Web API . . . . . . . . . . . . . . . . . . . . . . . . . | | 46 |
|  | 6.9 | Ciclul de viat,ă al unei cereri . . . . . . . . . . . . . . . . . . . | | 49 |
|  |  | 6.9.1 Hosting Layer . . . . . . . . . . . . . . . . . . . . . . . | | 50 |
|  |  | 6.9.2 | Message Handler Pipeline . . . . . . . . . . . . . . . . | 50 |
|  |  | 6.9.3 | Controller Handling . . . . . . . . . . . . . . . . . . . | 50 |
| 7 | Azure Cloud | | | 51 |

1. Prezentare generală a cloud computing . . . . . . . . . . . . . 51
2. Comparat,ie între local s,i Azure . . . . . . . . . . . . . . . . . 52

|  |  |
| --- | --- |
| 7.3 Ce oferă Cloud . . . . . . . . . . . . . . . . . . . . . . . . . . | 53 |
| 7.3.1 SaaS: Software as a Service . . . . . . . . . . . . . . . | 53 |

* 1. PaaS: Platform as a Service . . . . . . . . . . . . . . . 53
  2. IaaS: Infrastructure as a Service . . . . . . . . . . . . . 54

1. Servicii Azure . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

|  |  |
| --- | --- |
| 8 Arhitectura aplicat,iei | 55 |

1. Aplicat,ia client . . . . . . . . . . . . . . . . . . . . . . . . . . 55
2. Aplicat,ia desktop Java . . . . . . . . . . . . . . . . . . . . . . 55
3. Aplicat,ia server . . . . . . . . . . . . . . . . . . . . . . . . . . 56

|  |  |
| --- | --- |
| 9 Capitolul 8 | 57 |
| 9.1 PromON . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 57 |

|  |  |
| --- | --- |
| *CUPRINS* | 5 |
| Bibliografie | 63 |

6 *CUPRINS*

Capitolul 1

Introducere

1. Scurtă prezentare a aplicat,iei



Trăim într-o societate în care una dintre activităt,ile zilnice întreprinse de o persoană este vânatoarea de promot,ii. “Comorile” zilelor noastre, promot,iile, ne ajută deseori să reducem suma cheltuită pe un produs care ne era necesar s,i astfel să ne permitem să achizit,ionăm produse chiar cu un buget scăzut. După cum se s,tie, există s,i o „sărbătoare” în acest scop, s,i anume, „Black Friday”1, fiind extinsă deseori la o săptamână sau chiar mai mult, în care multe firme det,in o gamă largă de promot,ii, realizând în doar câteva zile, încasări record.

1traducere „Vinerea Neagră”

7

8 *CAPITOLUL 1. INTRODUCERE*

Aceasta este însă de multe ori doar o strategie de marketing, iar pro-dusele care par să aibă un pret, foarte avantajos să fie de fapt unul normal sau chiar mai ridicat ca în alte perioade ale anului. De aceea este indicat să căutăm promot,ii atunci cand chiar avem nevoie de un produs si să urmarim variat,ia pret,ului acestuia pe o perioadă de timp mai lungă. Promot,iile se desfăs,oară pe tot parcursul anului, chiar dacă nu sunt atat de mediatizate. Există promot,ii la aproape orice fel de produs iar căutarea acestora cu aju-torul motorului de căutare Google ne poate furniza atât informat,ii utile dar s,i informat,ii redundante, produse ale căror denumire cont,ine un cuvânt cheie atribuit atât produsului căutat dar s,i altor produse care pot să nu aibă nicio legătură cu căutarea dvs. De aceea m-am gândit la implementarea unei aplicat,ii care să grupeze laolaltă produsele promot,ionale de pe mai multe site-uri divizându-le în categorii sugestiv.

Consider că această aplicat,ie este de mare ajutor în zilele noastre, optimi-zând atât procesul de căutare, reducând durata acestuia, având o interfat,ă prietenoasă s,i us,or de utilizat de către oricine det,ine un smartphone sau o tabletă cu sistemul de operare Android.

1. De ce Android?

Am ales să dezvolt o aplicat,ie pe un dispozitiv mobil, întrucât smartphone-urile s,i tabletele devin din ce în ce mai populare s,i mai des utilizate, chiar dacă au apărut pe piat,a mondială relativ recent, primul model în 1992, nu-mit IBM Simon 2.

Conform datelor iSense.Solutions, platforma Android are un market share de peste 81% fiind lider de piat,ă atât în România cât s,i pe plan mondial. Exceptând Apple, care a ales iOS, s,i Microsoft, care utilizează sistemul de operare Windows Phone, aproape toate companiile producătoare de tele-foane mobile folosesc sistemul de operare al celor de la Google, Android.

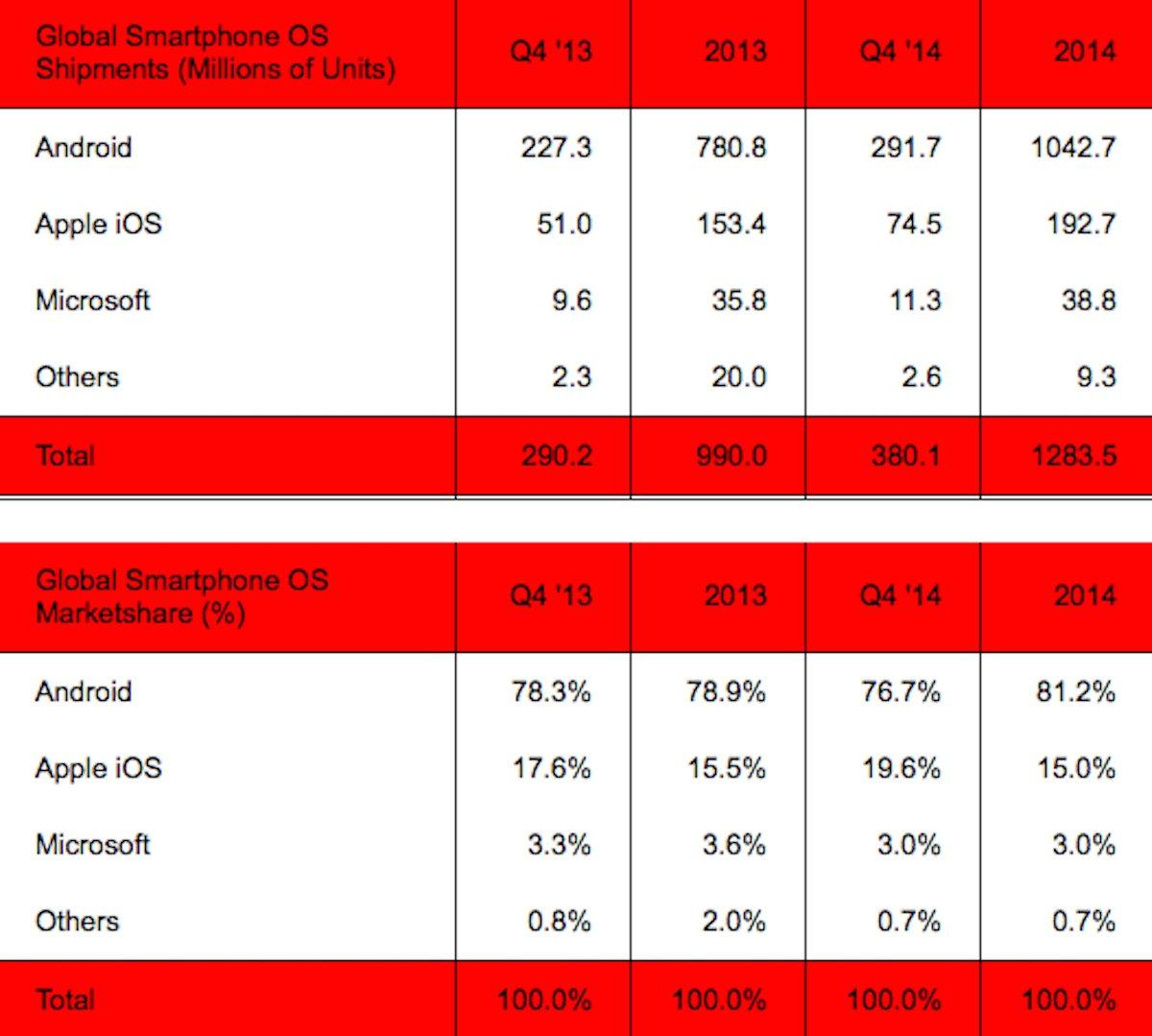
Astfel, consumatorii au de ales dintr-o arie foarte largă de telefoane având ca sistem de operare Android, s,i în acest fel pot să-s,i aleagă dintre toate mo-delele si dintre tot,i producătorii ceea ce li se potrives,te cel mai bine. Plat-forma Android se regăses,te de asemenea si pe tablete, ceasuri sau televizoare, numărul total de vânzari de smartphone-uri cu Android depăs,ind 1 miliard în 2014. 3

2http://www.qrcodescanning.com/smartphonehist.html

3http://www.businessinsider.com/android-1-billion-shipments-2014-strategy-analytics-2015-2

|  |  |
| --- | --- |
| *1.2. DE CE ANDROID?* | 9 |

Figura 1.1: Android vs alte sisteme de operare



Unul dintre cele mai cunoscute limbaje de programare este Java4. Dezvol-tat în anii 1990 de către Sun Microsystems, Java, la fel ca s,i platforma An-droid, se bucură de o popularitate crescută, aflându-se printre cele mai cău-tate limbaje de programare. Limbajul de programare Java este folosit atât pentru crearea cont,inutului paginilor web, aplicat,iilor desktop, a jocurilor, precum s,i pentru aplicat,iile pentru sistemul de operare Android. Aplicat,iile Java pot fi rulate pe mai multe platforme, de exemplu, un program scris în acest limbaj de programare poate fi folosit s,i pe sistemul de operare Windows s,i pe Mac OS X.

Am ales astfel să dezvolt o aplicat,ie având o temă actuală, utilizând ca mediu de programare platforma Android deoarece aceasta este cea mai uti-lizată la nivel mondial dar s,i din dorint,a de a învăt,a lucruri noi s,i de a-mi dezvolta cunos,tint,ele în această direct,ie având în vedere că tehnologia este în continuă dezvoltare iar eu ca programator trebuie să t,in pasul cu ea.

4mai multe detalii în Capitolul 4 - Limbajul de programare Java

10 *CAPITOLUL 1. INTRODUCERE*

Capitolul 2

Platforma Android

1. Istoric

Definition 2.1.1. *Android este o platformă software s,i un sistem de operare* *pentru dispozitive mobile precum smartphone-uri s,i tablete, bazată pe nucleul Linux, dezvoltată init,ial de compania Google, iar mai târziu de consort,iul comercial Open Handset Alliance[5].*

Android permite dezvoltatorilor să scrie cod gestionat în limbajul Java, controlând dispozitivul prin intermediul bibliotecilor Java dezvoltate de Google[6]. Aplicat,iile scrise în C s,i în alte limbaje pot fi compilate în cod mas,ină ARM

s,i executate, dar acest model de dezvoltare nu este sprijinit oficial de către Google[7][8].

Lansarea platformei Android în data de 5 noiembrie 2007 a fost anunt,ată prin fondarea Open Handset Alliance, un consort,iu de 48 de companii de hardware, software s,i de telecomunicat,ii, consacrat dezvoltării de standarde deschise pentru dispozitive mobile. Google a lansat cea mai mare parte a codului Android sub licent,a Apache, o licent,ă de tip free-software s,i open source.[5]

1. Arhitectura platformei

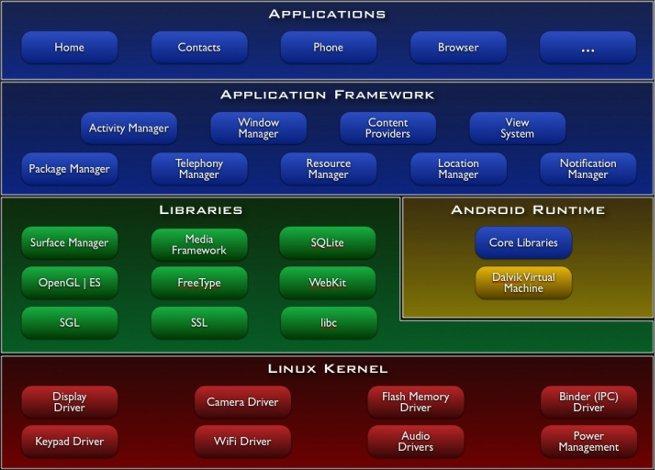
Android se bazează pe tehnologii variate open source pentru a pune la dispozit,ie o platformă mobilă solidă ce poate satisface toate necesităt,ile unui utilizator/dezvoltator.

Arhitectura platformei poate fi cel mai bine descrisă ca o serie de 5 niveluri principale, fiecare corespunzând unei operat,iuni de bază.

11

12 *CAPITOLUL 2. PLATFORMA ANDROID*

Figura 2.1: Arhitectura platformei



În imaginea anterioară se poate observa arhitectura platformei Android, care cuprinde cele 5 niveluri principale s,i subcomponentele acestora.[1]

1. Linux Kernel

Nivelul de bază al platformei Android este Linux kernel, acesta utilizându-se de binecunoscutul Linux kernel pentru a furniza funct,ionalitatea sistemu-lui de operare.

Pentru a satisface nevoile dispozitivelor mobile, nucleul Linux pe care se bazează Android a suferit schimbări arhitecturale. Cele mai notabile fiind cele ce vor fi prezentate în următoarea sect,iune.

|  |  |
| --- | --- |
| *2.3. LINUX KERNEL* | 13 |

Binder

Arhitectura platformei Android se foloses,te intens de comunicat,ia între pro-cese (inter-process communication-IPC). Aplicat,iile comunică cu sistemul, telefonul, serviciile s,i între ele utilizând IPC. Având în vedere că mecanis-mul IPC furnizat de sistemul de operare Linux nu este suficient, platforma Android a dezvoltat propriul sistem IPC, cunoscut sub numele de Binder, acesta fiind canalul central de comunicat,ie al întregii platforme Android.

Deoarece Binder este implementat ca un serviciu de bază in Linux kernel Android, dezvoltatorii de aplicat,ii nu interact,ionează direct cu acesta. Ma-joritatea API-urilor 1 frameworkului Android se bazează pe Binder pen-tru a interact,iona cu platforma într-un mod transparent dezvoltatorului de aplicat,ii.

Pentru a utiliza Binder în interact,ionarea cu alte aplicat,ii din sistem, SDK-ul Android 2 furnizează Android Interface Definition Language (AIDL). AIDL cuprinde funct,ionalitatea de a descompune obiectele primite în primitive ast-fel încât Binder le poate înt,elege s,i utiliza dincolo de limitele procesului.[2]

Logger

Logger este mecanismul esent,ial pentru depanare. Având în vedere faptul că aplicat,iile mobile depind mult de elemente din mediul înconjurător cum ar fi ret,elele WiFi s,i de datele care provin de la anumit,i senzori ai dispozitivului, funct,ionalităt,ile de logare ale aplicat,iei nu sunt îndeajuns de bune pentru probleme complexe. De aceea, este esent,ial să combinăm funct,ionalităt,ile de logare ale sistemului cu cele ale aplicat,iei pentru o imagine complexă.

Acest lucru devine mai greu de obt,inut atunci când dezvoltarea aplicat,iei s,i executarea acesteia au loc pe mas,ini diferite.

Android furnizează un sistem larg de servicii de logare care agreghează lo-gările care provin din partea platformei Android s,i cele care provin de la aplicat,iile care rulează pe aceasta.

SDK-ul Android-ului furnizează de asemenea uneltele necesare pentru a mo-nitoriza logările în timp real s,i suport pentru filtrare avansată.[2]

Wake Locks

Platforma Android este dezvoltată pentru a opera pe dispozitive mobile cu resurse limitate. Puterea bateriei este cea mai importantă s,i de aceea, dis-pozitivele Android intră in modul low-powered, numit s,i sleep. Chiar dacă acest mod permite utilizarea rezervei de baterie rămasă într-un mod eficient, nu este indicat ca un dispozitiv să intre în modul sleep pe parcursul unei operat,iuni importante ale unei aplicat,ii sau a unui serviciu.

Wake locks au fost introduse în Linux kernel al platformei Android cu scopul de a preveni aplicat,iile de la intrarea in modul sleep.[2]

1application programming interfaces

2Android Software Development Kit

14 *CAPITOLUL 2. PLATFORMA ANDROID*

Alarm Timer3

La fel cum am ment,ionat anterior în sect,iunea „Wake Locks”, dispozitivele intră în modul sleep pentru a conserva puterea bateriei. Pe parcursul mo-dului sleep, nicio aplicat,ie Android nu poate rula. Din acest motiv, a fost introdus alarm timer-ul care permite aplicat,iilor să programeze sarcinile de execut,ie. [2]

Low Memory Killer

La fel ca s,i bateria, memoria este de asemenea o resursă limitată a dispo-zitivelor mobile. Pe lângă dimensiunea memoriei, încărcarea în memorie a aplicat,iilor este de asemenea o operat,ie costisitoare. Pentru a rezolva aceasta problemă, platforma Android ment,ine toate aplicat,iile deschise încărcate în memorie chiar dacă utilizatorul nu mai interact,ionează cu ele.

Acest lucru permite utilizatorului o interschimbare rapidă între aplicat,ii. Această optimizare vine însă cu un cost: dispozitivul poate rămâne repede fără memorie dacă mai multe aplicat,ii sunt deschise.

Low Memory Killer cunoscut s,i ca Viking Killer, a fost introdus în Linux kernel al platformei Android pentru a administra s,i a revendica memoria înainte ca dispozitivul să rămână fără aceasta.

Atunci când memoria disponibilă scade sub un anumit prag, low memory killer, elimină din memorie aplicat,iile pornite, începând cu cele mai put,in importante.

Important,a unei aplicat,ii este definită după vizibilitatea acesteia pentru uti-lizator.

O aplicat,ie care rulează în prim plan este considerată cea mai importantă iar o aplicat,ie care rulează pe fundal nu este considerată importantă. Stările curente ale aplicat,iilor închise sunt salvate s,i pot fi reluate când utilizatorul se întoarce la o aplicat,ie.[2]

File System

Platforma Android se bazează pe Yet Another Flash File System (YAFFS2) ca s,i tip primar de sistem de fis,iere, fiind dezvoltat pentru a lucra pe NAND-based flash chips. Sistemul de fis,iere Android este de asemenea structu-rat într-un mod specific pentru a fi posibilă o actualizare us,oară a unor port,iuni ale platformei, aceste modificări neavând impact asupra celorlalte părt,i. Acest lucru este realizat prin ment,inerea port,iunilor platformei An-droid în partit,ii de sistem diferite, astfel garantându-se s,i o securitate mai ridicată întrucât componentele cheie ale platformei nu sunt mutabile pe par-cursul rulării, evitând infectarea cu virus,i s,i malware a componentelor cheie ale sistemului de operare.[2]

3în română, cronometru

|  |  |
| --- | --- |
| *2.4. NATIVE LIBRARIES* | 15 |

Partit,iile utilizate depind de producătorii dispozitivelor, cele mai des în-tâlnite fiind următoarele:

/boot /system

/recovery /data

/cache

1. Native Libraries

Pe nivelul superior nucleului Linux, platforma Android cont,ine un set de biblioteci native, precum:

SQLite: Furnizează o memorie internă ce relat,ionează cu o bază de date SQL pentru persistent,a datelor s,i accesarea rapidă a acestora într-un mod structurat

WebKit: Furnizează suport de interpretare HTML/CSS s,i un motor de executare JavaScript pentru a permite aplicat,iilor Android să în-corporeze tehnologii web

OpenGL ES: Furnizează funct,ionalităt,i performante pentru grafica 2D si 3D

Open Core: Furnizează un framework media care permite aplicat,iilor Android să înregistreze s,i să redea cont,inut audio s,i video

Open SSL: Furnizează protocoalele Secure Socket Layer (SSL) s,i Trans-port Level Security (TLS) care permit aplicat,iilor Android să comunice în mod securizat cu servicii la distant,ă prin utilizarea criptografiei

[1]

1. Android Runtimes

Android Runtime este port,iunea care dirijează platforma Android, rulând atât serviciile platformei cât s,i aplicat,iile care rulează pe platformă.

16 *CAPITOLUL 2. PLATFORMA ANDROID*

Android Runtime (ART)

Limbajul de programare oficial folosit pentru Android este Java. Java este un limbaj de programare obiect-orientat specific dezvoltării de aplicat,ii inde-pendente de platforma de lucru. Acest lucru este posibil deoarece codul Java compilat al aplicat,iei este realizat într-un limbaj independent de platformă numit bytecode. Acest bytecode este executat de Java Virtual Machine care rulează pe platformă.

The Android Runtime (ART) este noul Java Virtual Machine ce a fost introdus într-un mod experimental în versiunea de Android 4.4, ca mai apoi să fie introdus în versiunea Android 5.0 ca runtime-ul oficial al aplicat,iilor Android. Înainte de versiunea 5.0, aplicat,iile Android rulau pe Dalvik Vir-tual Machine (VM).

În comparat,ie cu abordarea Dalvik VM’s just-in-time (JIT) , ART se bazează pe o abordare ahead-of-time (AOT) , care permite compilarea codului byte-code s,i translatarea lui în cod mas,ină pe timpul instalării aplicat,iei. Acest lucru permite codului aplicat,iei să fie executat în viitor direct din mediul de rulare al dispozitivului.[1]

1. Android Applications

Compiled Android Applications

Chiar dacă oficial ART este runtime-ul utilizat pentru platforma Android începând cu versiunea 5.0, majoritatea dispozitivelor Android care rulează pe versiuni mai vechi de Android utilizează plaforma Dalvik VM.

Pentru a produce pachete de aplicat,ii care să fie compatibile atât cu ART cât s,i cu Dalvik VM, acestea sunt încă dezvoltate utilizând specificat,iile Dalvik. Deoarece a fost optimizat pentru dispozitive mobile, Dalvik VM procesează doar un tip special de bytecod numit Dalvik Executable (DEX). SDK-ul Androidului cuprinde unelte care pot traduce bytecode-ul standard din Java în bytecode DEX pe parcursul asamblării aplicat,iei android. Utilizarea bytecode-ului DEX furnizează multe avantaje în comparat,ie cu bytecode-ul standard Java. Art realizează conversia automată de la formatul Dalvik DEX la formatul ART OAT imediat după instalarea aplicat,iei pe dispozitiv.[1]

Application Sandbox

Platforma Android este construită având în vedere securitatea ca o necesitate importantă. Astfel rulează fiecare aplicat,ie într-un sandbox izolând datele aplicat,iei s,i codul executabil de alte aplicat,ii.

- Fiecare aplicat,ie rulează pe instant,a ART VM dedicată.

|  |  |
| --- | --- |
| *2.7. VERSIUNI ANDROID* | 17 |

* Datele aplicat,iei sunt protejate prin utilizarea unui fis,ier cu permisiuni ale sistemului. Fiecare aplicat,ie are asignat un cont la instalare s,i sistemul de operare restrict,ionează accesul la date pentru acel cont.
* Aplicat,iile pot să comunice cu sistemul sau cu alte aplicat,ii prin interfat,a de comunicare pe care platforma Android o pune la dispozit,ie. Aceste interfet,e sunt de asemenea protejate prin permisiuni Android[1].

Application Framework

Framework-ul aplicat,iei rulează pe ART VM s,i dispune de o interfat,ă pen-tru ca aplicat,iile Android să interact,ioneze cu platforma Android s,i cu dis-pozitivul mobil. Cuprinde servicii precum : Package Manager, Telephony Manager, Location Manager, and Notifications Manager.[1]

Applications

Spat,iul aplicat,iilor cuprinde toate aplicat,iile Android cu care se confruntă utilizatorul care rulează pe ART VM. Aceste aplicat,ii includ s,i aplicat,iile de la tert,i, care sunt downloadate din magazinul Android s,i aplicat,iile de sistem precum Aplicat,ia de pornire(Launcher), Contacte(Contacts), Tele-fon(Phone) s,i Browser. [1]

1. Versiuni Android



Dezvoltarea sistemul de operare Android a început în 2003 de Android Inc. care a fost apoi cumpărat de Google in 2005.[3] Cea mai recentă actualizare a sistemului Android este versiunea Android 5.0 „Lollipop”, care a fost lansată în 3 Noiembrie 2014. Începând cu Aprilie 2009, versiunile Android au fost denumite cu nume de deserturi, fiind lansate în ordine alfabetică, începand cu Android 1.5 “Cupcake”, versiunile anterioare 1.0 Alpha si 1.1 Beta fiind versiuni pre-comerciale.[9]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 18 | *CAPITOLUL 2.* | | | *PLATFORMA ANDROID* | | |
|  | Tabela 2.1: Versiuni Android | | | |  |  |
| Data lansării |  | Versiune | Nume | |  | Nivel |
|  |  |  |  |  |  | API |
| September 2008 |  | 1.0 | – |  |  | 1 |
| February 2009 |  | 1.1 | – |  |  | 2 |
| April 2009 |  | 1.5 | Cupcake | |  | 3 |
| September 2009 |  | 1.6 | Donut | |  | 4 |
| October 2009 |  | 2.0 | Éclair | |  | 5 |
| December 2009 |  | 2.0.1 | Éclair | |  | 6 |
| January 2010 |  | 2.1 | Éclair | |  | 7 |
| May 2010 |  | 2.2–2.2.3 | Froyo | |  | 8 |
| December 2010 |  | 2.3–2.3.2 | Gingerbread | |  | 9 |
| February 2011 |  | 2.3.3–2.3.7 | Gingerbread | |  | 10 |
| February 2011 |  | 3.0 | Honeycomb | |  | 11 |
| May 2011 |  | 3.1 | Honeycomb | |  | 12 |
| July 2011 |  | 3.2–3.2.6 | Honeycomb | |  | 13 |
| October 2011 |  | 4.0–4.0.2 | Ice | Cream | San- | 14 |
|  |  |  | dwich | |  |  |
| December 2011 |  | 4.0.3–4.0.4 | Ice | Cream | San- | 15 |
|  |  |  | dwich | |  |  |
| July 2012 |  | 4.1–4.1.2 | Jelly Bean | |  | 16 |
| November 2012 |  | 4.2–4.2.2 | Jelly Bean | |  | 17 |
| July 2013 |  | 4.3–4.3.1 | Jelly Bean | |  | 18 |
| October 2013 |  | 4.4–4.4.4 | KitKat | |  | 19 |
| July 2014 |  | 4.4w | KitKat with Wea- | | | 20 |
|  |  |  | rable Extensions | | |  |
| November 2014 |  | 5.0 | Lollipop | |  | 21 |

În 3 septembrie 2013, Google anuntă că un miliard de dispozitive Android activate sunt folosite în toată lumea.

În Ianuarie 2015, Android det,ine 62% din piat,a de smartphone-uri s,i tablete din US, 82.7% din piat,a chineză s,i 73.3% din piat,a europeană. [9]

|  |  |
| --- | --- |
| *2.8. ANDROID PLATFORM FRAGMENTATION* | 19 |

1. Android Platform Fragmentation

La o simplă vedere a datelor de lansare prezentate in tabelul 1-1, este posibil sa credet,i că cele mai multe dispozitive Android ruleaza pe minim versiunea Android 4.4, însă acest lucru nu este adevărat.

Datorită fragmentării mari, datele de lansare nu oferă o imagine clară a eco-sistemului Android.

Tabelul 1 este cel mai recent grafic de distribut,ie a versiunilor de Android Dashboard2 [2] pe baza datelor colectate în luna decembrie 2014.[1]

Tabela 2.2: Distribut,ia versiunilor Android

|  |  |  |  |
| --- | --- | --- | --- |
| Versiune | Nume versiune | Nivel API | Nivel distribut,ie |
| 2.2–2.2.3 | Froyo | 8 | 0.5% |
| 2.3.3–2.3.7 | Gingerbread | 9 | 9.1% |
| 4.0.3–4.0.4 | Ice Cream Sandwich | 15 | 7.8% |
| 4.1–4.1.2 | Jelly Bean | 16 | 21.3% |
| 4.2–4.2.2 | Jelly Bean | 17 | 20.4% |
| 4.3–4.3.1 | Jelly Bean | 18 | 7.0% |
| 4.4–4.4.4 | KitKat | 19 | 33.9% |

După cum se observă în tabelul 1-2, versiunea de Android 4.4 cuprinde doar sistemele de operare ale 33.9% de utilizatori din numărul total al aces-tora. Dezvoltarea unei aplicat,ii Android având ca target doar nivelul API 19 va limita audient,a la numai 33.9% dintre utilizatorii de bază Android.

Cu toate că un nivel API superior furnizeaza funct,ionalităti suplimentare, este de preferat sa luăm în calcul diagrama de distribut,ie pentru a determina numărul audient,ei.[1]

1. Android Support Library

Chiar dacă setarea unui target API mai mic măres,te numărul de persoane care vor putea utiliza aplicat,ia, de asemenea limitează funct,ionalităt,ile de care putem să ne folosim în dezvoltarea unei aplicat,ii Android. Pentru a remedia această problemă, a fost introdus Android Support Library. Acest pachet pune la dispozit,ie versiuni anterioare compatibile ale API-urilor re-cente.

Acest lucru inseamnă că o aplicat,ie care este dezvoltata pentru nivelul API 19 poate avea ca target nivelele API 4 s,i nivelurile următoare.

Includerea bibliotecii Android Support Library într-o aplicat,ie este conside-rată o tehnică „best practice”.[1]

20 *CAPITOLUL 2. PLATFORMA ANDROID*

Capitolul 3

Componetele unei aplicaţii Android

Framework-ul Android oferă o serie de componente pentru dezvoltarea de aplicaţii consistente şi interoperabile. În continuare vor fi prezentate componentele fundamentale oferite de platforma Android.[3]



1. Activity

„Activity” este numele dat pentru o aplicaţie în care utilizatorul poate interacţiona cu o singură fereastră la un moment dat. A fost ales numele de „activitate” pentru a sugera faptul că aplicaţia poate să realizeze o singură funcţionalitate focusată pe utilizator.

21

22 *CAPITOLUL 3. COMPONETELE UNEI APLICAŢII ANDROID*

Pe platforma Android, acest aspect de limitare al domeniului fiecarui ecran aduce funcţionalitea de interacţionare cu alte aplicaţii. De exemplu, la uti-lizarea e-mail-ului pe un dispozitiv mobil, când userul are posibilitatea de a selecta un contact din lista de contacte căruia să-i trimită e-mail-ul.

Acest lucru face posibilă reutilizarea codului şi menţinerea platformei in strânsă legătura.

Aproape toate aplicaţiile necesită comunicarea şi/sau interacţionarea cu un client, şi din acest motiv, activitatea se ocupă de crearea unei ferestre şi de aşezarea componentelor interfetei utilizator.[3]

1. Crearea unei activităţi

Pentru a crea o nouă activitate se derivează o clasa din android.app.Activity, după cum putem observa în exemplul 3.1.[3]

Example 3.1.1. *Crearea unei activităţi* *import android.app.Activity;*

*public class MyActivity extends Activity {*

*}*

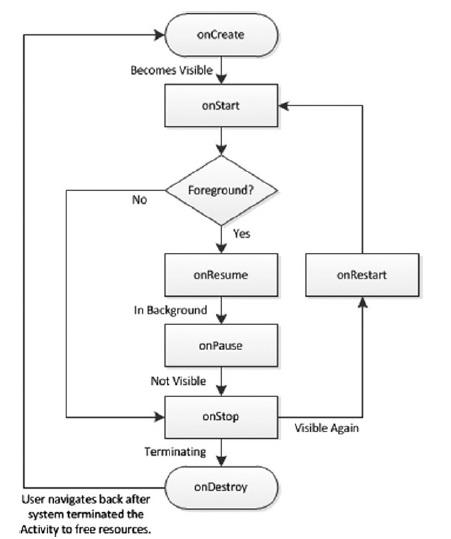
1. Declararea unei activităţi

Derivarea unei clase de acvitate nu înseamnă şi faptul că acea activitate este automat utilizabilă. Pentru ca o activitate să ia anumite roluri în aplica-tie, platforma Android necesită nişte meta-informaţii despre noua activitate. Aceste informaţii sunt furnizate de către fisierul XML Android manifest, unde eticheta XML /<activity/> declară noua activitate prin precizarea numelui de clasă, a titlului şi a modului în care va fi expusă utilizatorului.[1]

|  |  |
| --- | --- |
| *3.2. INTENT* | 23 |

1. Ciclul de viaţă al unei activităţi

[1]



1. Intent

După cum am menţionat şi anterior, platforma Android este dezvoltată pentru a fi extrem de modulară, promovând colaborarea dintre aplicaţiile prezente pe dispozitiv. Astfel, aplicaţiile pot îmbina activităţi pentru a oferi

˘

o anumită functionalitate utilizatorului. Acest lucru este posibil printr-un binding runtime întârziat, cunoscut sub numele de android.content.Intent, pe care platforma Android îl pune la dispoziţie.

Intent-ul deţine o structură pasivă de date care memorează o descriere abs-tractă a acţiunii ce urmează a fi întreprinsă. Principalele informaţii dintr-un intent sunt:

action, care reprezintă în general acţiunea ce va fi întreprinsă

data, care reprezintă un obiect URI (uniform resource identifier) ce se referă la datele care vor fi prelucrate

Cea mai importantă utilizare a intentului este de a lansa activităţile si ser-viciile.

24 *CAPITOLUL 3. COMPONETELE UNEI APLICAŢII ANDROID*

1. Service

Pentru aplicaţii cu durată de rulare mai mare, durata de viaţă a unei ac-tivităţi s-ar putea să nu fie indeajuns de lungă, de exemplu daca o aplicaţie necesită descărcarea de pe Internet a unui videoclip, platforma va termina activitatea imediat după ce utilizatorul păraseste temporar aplicaţia fără a verifica dacă operaţia s-a incheiat sau nu.

Framework-ul Android furnizează o componentă a aplicaţiei, numită an-droid.app.Service, pentru a permite aplicaţiilor să execute operaţii cu durrata de rulare mai mare în planul secundar.

Serviciile pot fi utilizate şi din aceeasi aplicaţie dar şi de alte aplicaţii de pe dispozitiv dacă serviciul este exportat.[1]

1. Crearea unui Service

Crearea unui serviciu se face prin simpla derivare a unei noi clase din android.app.Service cum se vede în exemplul 3.2.[2]

Example 3.3.1. *Crearea unui Service import android.app.Service;* *public class MyService extends Service {*

*}*

1. Declararea unui Service

Crearea unui noi serviciu nu îl face imediat utilizabil pe platforma An-droid. Aceasta are nevoie de meta-informaţii pentru a putea utiliza serviciul.

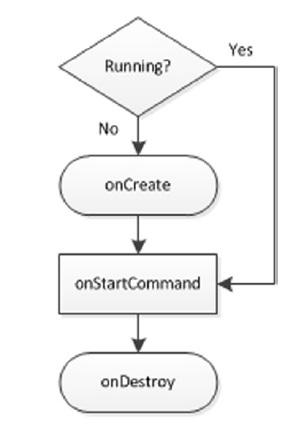
Aceste meta-informaţii se găsesc , de asemnea în fişierul Android manifest

ˆ

fiind marcate cu tagul XML <service>.

Asemenea elementului <activity>, elementul <service> poate să conţina ele-mente <intent-filter> în corpul său pentru a specifica ce acţiuni să indeplinească.[2]

1. Ciclul de viaţă al unui Service



|  |  |
| --- | --- |
| *3.4. CONTENT PROVIDER* | 25 |

1. Content Provider

După cum am menţionat anterior în secţiunea „Service”, atât activită-ţile cât şi serviciile pot fi exportate astfel încat alte aplicatii care rulează pe acelaşi dispozitiv mobil să poată interacţiona simultan cu acestea. Chiar dacă accesarea unor părţi ale altor aplicaţii poate fi foarte utilă, în anumite cazuri, este nevoie de simpla accesare a datelor, fără accesarea anterioară a activităţilor si a serviciului.

Platforma Android furnizeaza o componentă aplicaţiei, numită content pro-vider, care manageriază accesul la un set structurat de date. Este o interfaţă standard care conectează datele unei aplicaţii cu codul executabil al altei aplicaţii.[2]

1. Crearea unui Content Provider

Un nou content provider poate fi creat prin simpla derivare a unei noi clase din android.content.ContentProvider ca în exemplul 3.3.

Example 3.4.1. *Crearea unui Content Provider import android.content.ContentProvider;* *public class MyContentProvider extends ContentProvider {*

*}*

Clasa abstractă ContentProvider cuprinde 6 metode abstracte care nece-sită implementare.[2]

query - metodă apelată pentru a interoga providerul pentru date insert - metodă apelată pentru a insera continut nou in provider

update - metodă apelată pentru a actualiza continutul unui provider cu un nou cont,inut

delete- metodă apelată pentru a sterge providerul

getType -metodă apelată pentru a prelua tipul MIME pentru cont,inutul ce va fi returnat pentru URI-ul dat

onCreate - metodă apelată de platformă când se instanţiază provide-rul.(apelată inainte de orice altă metodă)

26 *CAPITOLUL 3. COMPONETELE UNEI APLICAŢII ANDROID*

1. Broadcast Messages

Platforma Android furnizează un sistem larg de mesagerie numit broad-cast messages. Această facilitate permite aplicaţiilor şi sistemului să propage evenimente şi schimbări de stare pentru parţile interesate prin emiterea unui intent ca mesaj.

Un mesaj broadcast poate fi trimis prin metoda sendBroadcast.

Pentru receptarea unui mesaj broadcast, aplicaţia trebuie să deriveze o nouă clasă din android.content.BroadcastReceiver.

Simpla derivare a clasei nu este insă suficientă pentru a recepţiona me-sajele. Aplicaţia trebuie să informeze platforma Android dinamic prin cod sau prin înregistrarea în fisierul manifest privire la intent,ia de a recepţiona mesaje broadcast.

Tagul XML <receiver> este utilizat pentru a înregistra un receptor broad-cast în manifestul aplicaţiei.

Pentru înregistrarea dinamică prin cod, există metoda registerReceiver.[1]

Capitolul 4

Limbajul de programare Java

1. Platforma Java

Definition 4.1.1. *Java este un limbaj de programare orientat-obiect, pu-ternic tipizat, conceput de către James Gosling la Sun Microsystems (acum filială Oracle) la începutul anilor ’90, fiind lansat în 1995. Cele mai multe aplicat,ii distribuite sunt scrise în Java, iar noile evolut,ii tehnologice permit utilizarea sa s,i pe dispozitive mobile gen telefon, agenda electronică, palmtop etc. În felul acesta se creează o platformă unică, la nivelul programatorului, deasupra unui mediu eterogen extrem de diversificat. Acesta este utilizat în prezent cu succes s,i pentru programarea aplicat,iilor destinate intranet-urilor. [12]*

Java este un limbaj de programare utilizat în toate industriile pentru aproape orice tip de aplicat,ie. Există aproximativ 6 milioane de programatori Java în lume, însă sunt de asemenea s,i multe posturi disponibile, astfel că un programator care are cunos,tint,e medii de programare Java are mai multe s,anse să-si gasească un job fat,ă de un programator expert în alte limbare de programare precum C/C++, Ruby, etc.

Java a apărut pe scena tehnologiei în anul 1995 s,i a devenit popular foarte repede. Acest lucru s-a întâmplat în parte din cauza JVM. JVM este ca un translator de limbi străine, având loc transformarea bytecode-ului Java în orice limbaj nativ ce poate fi interpretat de un anumit calculator.[10] Programele Java sunt rulate (sau interpretate) de către un alt program numit Java VM. Programele nu rulează direct pe sistemul de operare nativ, ci sunt interpretate de Java VM pentru sistemul de operare nativ.

27

28 *CAPITOLUL 4. LIMBAJUL DE PROGRAMARE JAVA*

Acest lucru înseamnă că orice sistem de operare cu Java VM instalat poate rula programe Java, indiferent de sistemul de operare pe care au fost dezvoltate init,ial aplicat,iile.

Aceasta se numes,te portabilitate, s,i în lumea programării, portabilitatea este un produs foarte pret,ios.[11]

Platforma Java constă în API-ul Java s,i mas,ina virtuală Java (JVM).

API-urile Java sunt biblioteci de cod compilat care se folosesc în dezvol-tarea programelor java. Acestea permit adăugarea unor funct,ionalităt,ii per-sonalizabile pentru a economisi timp de programare, unul dintre cele mai simple exemple fiind reprezentat de API-ul folosit pentru a printa o linie de cod în consolă; comanda pentru printare în consolă este furnizată din API s,i poate fi folosită numai prin specificarea textului ce va fi printat.

De exemplu, un program Java dezvoltat pe un computer personal (PC) cu sistemul de operare Windows NT ar trebui să ruleze la fel de bine fără modificări pe o stat,ie de lucru cu sistem de operare Linux, s,i vice-versa.[10]

1. Configurarea unui computer

Pentru a scrie programe Java, sunt necesare:

un compilator Java un JVM

API-ul Java

acces la documentat,ia API-ului Java

un editor pentru redactarea programelor Java.

un program care să interpreteze s,i să execute comenzile redactate

Pentru ultimele două act,iuni se foloses,te, în general, o singură interfat,ă pri-etenoasă, numită mediu de dezvoltare integrat(integrated development en-vironment - IDE). Printre cele mai cunoscute IDE - uri utilizate pentru dezvoltarea de aplicat,ii Java se numără: Eclipse, NetBeans, IntelliJ IDEA, JDeveloper, JCreator, etc.

|  |  |
| --- | --- |
| *4.3. LIMBAJ DE PROGRAMARE OBIECT ORIENTAT* | 29 |

În procesul de realizare a aplicat,iei mele, am utilizat pentru programul de citire a link-urilor s,i actualizarea periodică a bazei de date, ca s,i mediu de dezvoltare integrat Eclipse IDE, deoarece:

poate fi descărcat s,i apoi utilizat gratuit

este în topul celor mai utilizate Java IDE-uri de către programatorii consacrat,i

este intuitiv s,i us,or de utilizat

Eclipse nu este folosit doar pentru dezvoltarea unei aplicat,ii Java, ci poate fi folosit împreună cu o serie de alte limbaje de programare (Ada, ABAP, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Lua, Natural, Perl, PHP, Prolog, Python, R, Ruby, Scala, Clojure, Groovy, Scheme s,i Erlang), acest lucru reprezentând un plus semnificativ al acestui IDE.

JDK s,i JRE

Pentru dezvoltarea de programe Java pe un computer este nevoie să fie in-stalat JDK. Pentru rularea de programe Java care au fost compilate deja pe un alt computer, este de ajuns Java Runtime Environment (JRE). JDK include si JRE.

1. Limbaj de programare obiect orientat

Java este un limbaj de programare obiect orientat, ceea ce înseamnă că se pot construi si reprezenta obiecte din lumea reală. Fiecare program Java are cel putin o clasă care are anumite funct,ionalităt,i. De exemplu, cea mai simplă clasă, HelloWorld, are funct,ionalitatea de a saluta lumea. Clasele în Java pot să cont,ină metode(funct,ii) s,i atribute (proprietăt,i).

Principiile programării obiect orientate

1. Abstractizarea datelor

Abstractizarea datelor reprezintă procesul de definire a unui tip de date denumit tip abstract de date(abstract data type-ADT), recurgând s,i la as-cunderea datelor. Definirea unui tip abstract de date implică specificarea reprezentării interne a datelor pentru acel tip, precum s,i un set suficient de funct,ii cu ajutorul cărora putem utiliza acel tip fără a fi necesară cunoas,terea structurii sale interne. [15]

30 *CAPITOLUL 4. LIMBAJUL DE PROGRAMARE JAVA*

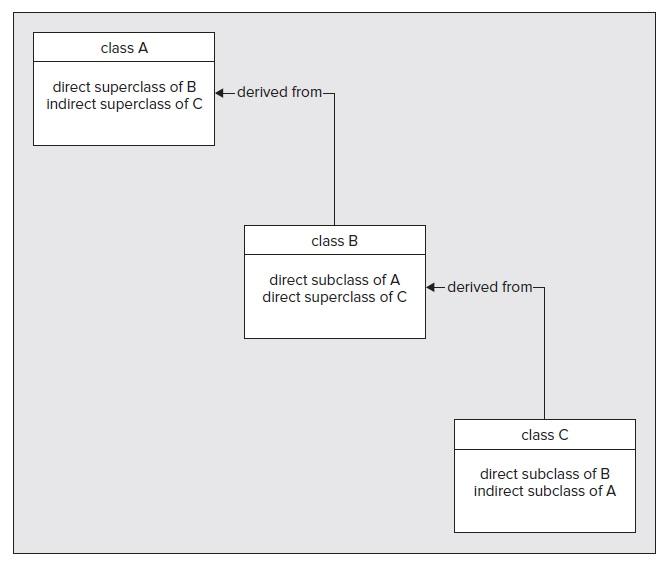
1. Încapsularea

Încapsularea se referă la ascunderea datelor s,i a metodelor într-un obiect., prin specificarea acestora ca fiind private în definit,ia clasei. Astfel, accesul la date se face doar prin intermediul metodelor descrise în clasă. Încapsu-larea este importantă pentru a asigura securitatea s,i integritatea obiectelor unei clase. Un mare avantaj al încapsulării este ascunderea implementării clasei astfel că metodele interne ale clasei pot fi modificate după cum vrea programatorul s,i atâta timp cât funct,ionalitatea externă nu este modificată, utilizatorul clasei nu va observa nicio diferent,ă la utilizarea metodei.[14]

1. Mos,tenirea

În limbajele de programare obiect-orientate, termenul de mos,tenire se referă la abilitatea de a defini o nouă clasă bazată pe una care deja există. Clasa utilizată ca s,i clasă de bază poate fi definită de aceeas,i persoană, poate fi preluată de la altcineva sau dintr-un pachet Java specific. Clasa derivată din clasa de bază se mai numes,te subclasă, clasă copil iar cea de bază se mai numes,te s,i superclasă, clasă părinte. Clasa derivată va mos,teni de la superclasa sa toate funct,ionalităt,ile pe care aceasta le are dar poate să cont,ină s,i funct,ionalităt,i noi sau să le suprascrie pe cele existente.[14]

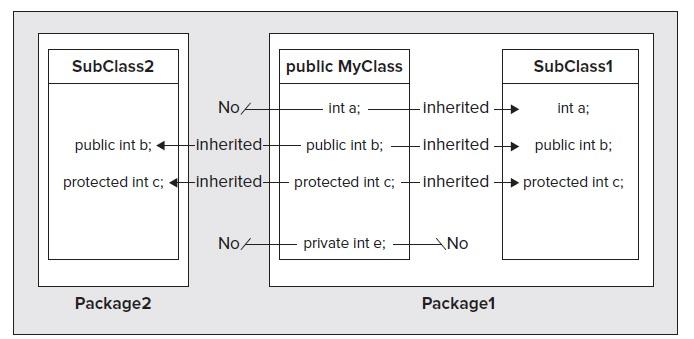
În figura următoare se observă o ierarhie de clase realizată prin mos,tenire:



|  |  |
| --- | --- |
| *4.3. LIMBAJ DE PROGRAMARE OBIECT ORIENTAT* | 31 |

După cum putet,i vedea în Figura , o subclasă care se află în acelas,i pachet ca s,i clasa de bază mos,tenes,te totul cu except,ia membrilor privat,i ai clasei de bază. Dacă se defines,te o subclasă în afara pachetului ce cont,ine clasa de bază, atributele private nu sunt mostenite s,i nici ceilalt,i membrii din clasa de bază care au fost declarat,i fără specificator de acces sau cei declarat,i ca private.

Astfel, specificatorul public este cel mai put,in restrictiv, fiind vizibil ori-



unde, protected este următorul care previne accesul din clasele aflate în afara pachetului dar nu limitează mos,tenirea, clasa derivată poate să accese mem-brii clasei de bază declarat,i protected. Cel mai restrictiv specificator este private, accesul fiind valabil doar în interiorul clasei în care a fost declarat mebrul privat.

1. Suprascrierea unei metode

Un alt termen important din programarea obiect-orientată este repre-zentat de către suprascrierea metodelor. Putem defini o metodă într-o clasă derivată, care are aceeas,i semnătură ca s,i o metodă în clasa de bază. Având aceeas,i semnătură înseamnă că numele metodei trebuie să fie acelas,i s,i lis-tele de parametri trebuie să cont,ină acelas,i număr de parametri cu tipuri identice. Atributul de acces pentru metoda în clasa derivată poate fi acelas,i cu cel din clasa de bază sau poate fi mai put,in restrictiv, dar nu poate fi mai restrictiv. Acest lucru înseamnă că, dacă declaram o metodă publică în clasa de baza, orice proprietate definită în clasa derivată trebuie să fie, de asemenea, declarată ca publică. Nu se poate omite atributul de acces în clasa derivată în acest caz, sau specificarea aceastuia ca fiind private sau protected. Când se defines,te o nouă versiune a unei metode din clasa de bază în acest fel, metoda din clasă derivată suprascrie metoda din clasa de baz. [14]

32 *CAPITOLUL 4. LIMBAJUL DE PROGRAMARE JAVA*

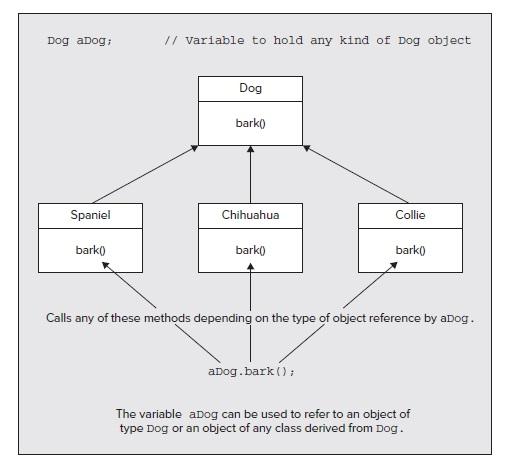
1. Adnotarea @Overriding

Când se defines,te o metodă într-o clasă derivată, care este destinată pen-tru a suprascrie o metodă din superclasă, este us,or să se facă o gres,eală în semnătura pentru metoda de clasă derivată. Dacă numele s,i lista de pa-rametri din metoda clasei derivate nu sunt identice cu cele ale metodei din superclasa, are loc supraîncărcarea metodei din clasa de bază s,i nu suprascri-erea acesteia. Pentru evitarea acestui lucru s,i pentru o mai bună lizibilitate a codului scris se foloses,te adnotarea @Overriding înainte de metoda care va suprascrie metoda din clasa de bază. Astfel compilatorul este informat s,i în caz că nu sunt indeplinite toate condit,iile de suprascriere se va semnala o eroare de compilare.[14]

1. Polimorfismul

Polimosfismul înseamnă, în general, abilitatea de a avea mai multe forme. În programare, reprezintă abilitatea unei singure variabile de un tip dat să fie utilizată pentru a referi obiecte de tipuri diferite s,i să apeleze automat metoda specifică tipului obiectului referit. Aceasta presupune faptul că o singură metodă poate să se comporte diferit, în funct,ie de obiectul apelat.

Acest lucru este evident,iat în figura următoare:[14]



|  |  |
| --- | --- |
| *4.3. LIMBAJ DE PROGRAMARE OBIECT ORIENTAT* | 33 |

Condit,iile care trebuiesc îndeplinite pentru a utiliza polimorfismul sunt următoarele:[14]

apelul metodei din clasa derivată trebuie să se facă prin intermediul unei variabile de tipul clasei de bază

metoda apelată trebuie să fie definită în clasa de derivată

metoda apelată trebuie să fie, de asemenea, declarată ca membru al clasei de bază

semnătura metodei din clasa de bază trebuie să fie la fel cu semnătura metodei din clasa derivată sau tipurile returnate să fie covariante

specificatorul de acces din clasa derivată nu trebuie să fie mai restrictiv ca specificatorul de acces din clasa de bază

1. Clasa de bază universală

Toate clasele definite în Java sunt implicit subclase ale clasei Object, nefiind nevoie să fie specificat acest lucru. Astfel, cu ajutorul clasei Object, putem să scriem metode care vor utiliza obiecte de tipuri necunoscute, putem să definim parametri de tip Object, în acest fel metoda va putea fi folosită pentru orice tip de obiect trimis ca parametru. De asemenea, clasa Object cuprinde 7 metode publice s,i 2 protejate, pe care le vom vedea în tabelul următor:[14]

|  |  |  |
| --- | --- | --- |
| 34 | *CAPITOLUL 4. LIMBAJUL DE PROGRAMARE JAVA* | |
|  | Tabela 4.1: Metodele publice ale clasei Object | |
| Metodă |  | Scopul metodei |
| toString() |  | Această metodă returnează un obiect String care |
|  |  | descrie obiectul curent. În versiunea mos,tenită |
|  |  | a acestei metode va fi returnat numele clasei ur- |
|  |  | mat de ‘@’ s,i de reprezentarea hexazecimală a |
|  |  | obiectului. Această metodă este apelată auto- |
|  |  | mat atunci când se concatenează obiecte cu va- |
|  |  | riabile String utilizând operatorul +. Această |
|  |  | metodă poate fi suprascrisă într-o clasă, pentru |
|  |  | a crea un string personalizat pentru obiectul cla- |
|  |  | sei. |
| equals() |  | Această metodă compară referint,a la obiectul |
|  |  | trimis ca argument cu referint,a la obiectul curent |
|  |  | s,i returnează true, dacă acestea sunt egale. True |
|  |  | este returnat dacă obiectul curent s,i argumen- |
|  |  | tul reprezintă acelas,i obiect, nu au doar valorile |
|  |  | egale iar false este returnat dacă sunt obiecte di- |
|  |  | ferite chiar s,i dacă atributele acestora au valori |
|  |  | identice. |
| getClass() |  | Această metodă returnează un obiect de tipul |
|  |  | Class care identifică clasa obiectului curent. |
| hashCode() |  | Această metodă calculează valoarea hashcode |
|  |  | pentru un obiect s,i o returnează ca int. Valo- |
|  |  | rile hashcode sunt utilizate in clasele definite în |
|  |  | pachetul java.util pentru a memora obiectele în |
|  |  | tabele de dispersie (tabele hash). |
| notify() |  | Această metodă este folosită pentru a "‘trezi"’ |
|  |  | un fir de execut,ie asociat cu obiectul curent. |
| notifyAll() |  | Această metodă este folosită pentru a "‘trezi"’ |
|  |  | toate firele de execut,ie asociat cu obiectul cu- |
|  |  | rent. |
| wait() |  | Această metodă este folosită pentru a determina |
|  |  | un fir de execut,ie să as,tepte o schimbare în obiec- |
|  |  | tul curent. |

De ret,inut că metodele getClass(), notify(), notifyAll() s,i wait() nu pot fi suprascrise într-o clasă definită de noi întrucât sunt specificate ca fiind finale în definit,ia clasei Object.

|  |  |
| --- | --- |
| *4.3. LIMBAJ DE PROGRAMARE OBIECT ORIENTAT* | 35 |

Pentru metoda toString() este sugerat să utilizăm mereu adnotarea @Over-ride în clasa noastră pentru a nu se folosi polimorfismul prin apelarea metodei din clasa Object.

Cele două metode protected ale clasei Object sunt:

|  |  |  |
| --- | --- | --- |
|  | Tabela 4.2: Metodele protejate ale clasei Object | |
| Metodă |  | Scopul metodei |
| clone() |  | Această metodă creează un obiect care este o |
|  |  | copie a obiectului curent fără să t,ină cont de |
|  |  | tipul acestuia, poate fi de orice tip. Acest lucru |
|  |  | poate fi realizat doar dacă obiectul ce urmează a |
|  |  | fi clonat indică faptul că este acceptată clonarea |
|  |  | sa prin implementarea interfet,ei Cloneable. |
| finalize() |  | Această metodă poate fi apelată pentru a |
|  |  | „curăt,a” după ce un obiect a fost distrus. |

Utilizarea metodei clone() pentru a duplica obiectele poate fi complicată s,i dacă nu se realizează corect poate să ducă la rezultate neas,teptate. De aceea pentru a clona obiecte ale clasei definite de noi este indicat să utilizăm un constructor de copiere.

36 *CAPITOLUL 4. LIMBAJUL DE PROGRAMARE JAVA*

Capitolul 5

JSOUP API

Jsoup este o librărie Java pentru lucrul cu elemente specifice HTML, cuprinzând metode speciale pentru a manipula datele utilizând cele mai po-pulare metode DOM, CSS si jquery.

Jsoup :

găses,te s,i extrage date utilizâd DOM1 traversal sau selectori CSS2

parsează pagini HTML prin intermediul URL-ului, a unui fis,ier HTML sau a unui string.

manipulează elementele HTML, atributele s,i textul.

curăt,ă cont,inutul prezentat de utilizatori utilizănd o listă albă pentru a preveni XSS3

datele de ies,ire reprezintă HTML ordonat

jsoup este proiectat să se ocupe de toate tipurile de HTML care există; de la curat s,i valid, la taguri-soup invalide; jsoup va crea un arbore de parsare sensibil.

1Document Object Model (DOM) este o convent,ie cross-platform s,i independentă de limbajul utilizat, folosită pentru reprezentarea s,i interact,ionarea cu obiecte în documente HTML, XHTML, s,i XML.[16]

2Cascading Style Sheets (CSS) este un limbaj de stil utilizat pentru design-ul s,i for-matarea unui document scris într-un limbaj markup.

3Cross-site scripting (XSS) este un tip de vulnerabilitate a securităt,ii unui computer, în general găsită în aplicat,iile Web.

37

38 *CAPITOLUL 5. JSOUP API*

Example 5.0.1. *Exemplu de utilizare a API-ului jsoup*

*Descarcă pagina Wikipedia, o parseaza ca un DOM s,i selectează titlurile din sectiunea de stiri intr-o lista de Elements:*

*Document doc = Jsoup.connect("http://en.wikipedia.org/").get(); Elements newsHeadlines = doc.select("#mp-itn b a");*

Open source

Jsoup este un proiect open source distribuit sub licent,a MIT4. Codul sursă este disponibil pe GitHub.

Pentru a utiliza jsoup, trebuie descărcată arhiva jar utilizând url-ul ht-tps://jsoup.org/download, versiunea 1.8.2.

Aceasta trebuie dezarhivată s,i importată în proiectul în care umrează să fie folosit.

Dacă se utilizează Maven 5 pentru a manageria dependent,ele dintr-un proiect Java(lucru recomandat), nu este nevoie să descărcăm arhiva jar, ci este suficient să punem următoarea bucată de cod în fis,ierul POM, sect,iunea <dependencies>:

<dependency>

<!– jsoup HTML parser library @ http://jsoup.org/ –> <groupId>org.jsoup</groupId> <artifactId>jsoup</artifactId> <version>1.8.2</version>

</dependency>

Dependint,e

Jsoup nu are dependint,e.

Jsoup rulează cu Java începând de la versiunea 1.5, cu Scala, Android, OSGi s,i Google App Engine.

Pentru a învăt,a cum se utilizează librăria jsoup este recomandat să par-curgem Cookbook-ul care se găses,te pe https://jsoup.org/cookbook/ care cuprinde:[18]

Introducere

1. Parsarea s,i parcurgerea unui Document

4Licent,a MIT este o licent,ă gratuită având originea la Insitutul de Tehnologie din Massachusetts [17]

5Maven este un instrument automat utilizat în general în proiectele Java. Maven descarcă dinamic librării Java s,i plug-in-uri Maven din unul sau mai multe depozite precum Maven 2 Central Repository, s,i le stochează în cache-ul local.

39

Input

1. Parsarea unui document preluat dintr-un String
2. Încărcarea unui Document printr-un URL
3. Încărcarea unui Document dintr-un fis,ier

Extragerea de date

1. Utilizarea de metode DOM pentru a naviga într-un document
2. Utilizarea de selectori sintactici pentru a căuta elemente
3. Extragerea de atribute, text s,i HTML din elemente
4. Lucrul cu URL-uri
5. Un exemplu de program - printarea unor link-uri

Modificarea datelor

1. Setarea valorilor unor atribute
2. Setarea HTML pentru un element
3. Setarea cont,inutului text al elementelor

Curăt,area HTML

1. Eliminarea surselor HTML nesigure pentru prevenirea XSS

Try jsoup

Try jsoup este un demo online, interactiv care permite vizualizarea parsării unui HTML într-un DOM s,i testarea interogărilor CSS.

40 *CAPITOLUL 5. JSOUP API*

Capitolul 6

ASP.NET Web API

1. REST

Putem defini Representational State Transfer (REST), ca un stil arhitectural situat în partea superioară a unei serii de principii. Cres,terea REST în ultimii ani este legată de design-ul API-ului, pe care foarte multe aplicat,ii web îl oferă pentru a extinde funct,ionalităt,ile lor. Chiar dacă nu este legat de HTTP, REST este în general asociat cu aplicat,ii web. Se întâmplă ca HTTP să se potrivivească foarte bine cu principiile REST.[19]

Principiile REST sunt:

Uniform Interface (interfat,ă uniformă) Stateless (fără stare)

Cacheable (se poate salva într-un cache) Client-Server

Layered System(sistem stratificat) Code on Demand (cod la cerere)

Ideea de REST folosit pe HTTP este de a utiliza funct,ionalitatea proto-colului cât mai mult posibil, astfel încât să nu se reinventeze roata.

1. Uniform Interface

În „centrul" REST sunt resursele, acele „lucruri" ce se doresc a fi gestio-nate utilizând API-ul. O resursă poate fi o postare pe blog, un client, un document, s,i, în general, tot ceea ce se dores,te să fie expus. O resursă are un identificator, as,a cum o înregistrare în baza de date are o cheie primară. În acelas,i fel, o resursă are un URI care identifică resursa.URI-ul nu este o reprezentare a resursei, care poate avea diferite formate. Acesta este doar un identificator ce poate fi folosit pentru a accesa resursa.

41

42 *CAPITOLUL 6. ASP.NET WEB API*

O resursă poate fi solicitată (requested) folosind URI-ul, s,i ceea ce se obt,ine este o reprezentare a acestei resurse într-un anumit format. Formatul este negociat între client s,i server s,i ar putea fi orice, de la XML s,i JSON, până la HTML, PNG, CSV sau alte formate binare. Cu reprezentarea resur-sei, clientul poate manipula starea s,i poate lucra cu acea resursă, utilizând serverul, dar doar dacă are drepturi să facă acest lucru.[19]

1. Stateless

Stateless este un principiu fundamental pentru o aplicat,ie REST; serverul nu ar trebui să stocheze informat,ii despre client,i. Acest lucru înseamnă că, atunci când o cerere ajunge la server, serverul încarcă resursa din spat,iul de stocare (de obicei o bază de date) s,i trimite înapoi reprezentarea resursei către client. Aceasta este starea resurselor. Dacă o secundă mai târziu starea resursei în spat,iul de stocare se schimbă datorită unei noi cereri, clientul nu ar trebui să s,tie de această modificare.

Stateless înseamnă, de asemenea, că serverul nu ar trebui să folosească sesiuni sau alte mecanisme pentru a stoca informat,ii despre client, iar fiecare cerere nu trebuie să fie corelată cu cererile trecute sau viitoare.[19]

1. Cache-able

Clientul poate „cache-ui" resursa, iar serverul trebuie să ofere informat,ii re-feritoare la capabilitatea de cache a resursei. Dacă o resursă este „cache-uită" corespunzător, atunci se poate mics,ora numărul de cereri către server.[19]

1. Client-server

Singurele lucruri pe care clientul le vede sunt URI-ul s,i reprezentarea re-sursei. Clientul nu poate vedea (s,i cu sigurant,ă nu este interesat în a vedea), unde sunt stocate resursele. Pe de altă parte, serverul nu trebuie să s,tie dacă clientul are o anumită resursă, iar dacă interfat,a nu se schimbă, cererile între server s,i client pot fi efectuate fără a fi vreo problemă. [19]

1. Layered System

Clientul s,tie foarte put,ine lucruri despre server; nu s,tie, de exemplu dacă este direct conectat la server, sau dacă a ajuns la server trecând printr-un proxy sau alt server intermediar (balancer, etc).[19]

|  |  |
| --- | --- |
| *6.7. CODE ON DEMAND* | 43 |

1. Code on Demand

Serverul poate extinde funct,ionalitatea clientului prin transmiterea de cod executabil. De exemplu, un server poate trimite JavaScript către client, care poate face un anumit tip de operat,ie asupra datelor. Unul din punctele cheie al REST este scalabilitatea. Faptul că serverul nu trebuie să stocheze informat,ii despre client ajută la salvarea memoriei. Sistemul stratificat per-mite utilizarea de servere cache ca s,i load-balancer pentru a obt,ine scalabi-litate. Adăugarea de noi servere atât timp cât se respectă principiile client-server, permite schimbări de implementare (de exemplu, se poate trece de la o bază de date SQL la una NoSQL) fără s,tirea clientului.

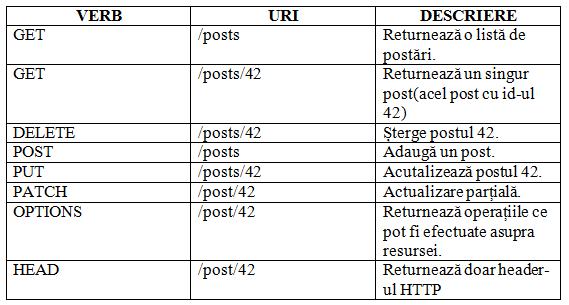
Dar cum se obt,ine acest lucru s,i cum funct,ionează? În majoritatea arti-colelor, arhitectura REST nu este legată de HTTP, dar HTTP pare perfect pentru a construi un API REST, din moment ce majoritatea lucrurilor pe care REST le defines,te sunt deja construite chiar în protocol (capabilitatea de a cache-ui, de exemplu).

Web-ul în sine este REST: URL-ul este identificatorul paginii ce se doress,te a fi accesată, se introduce URL-ul în browser pentru a obt,ine o reprezentare în format HTML, s,i se foloses,te un link pentru a transfera starea la o altă pa-gină. Un aspect al REST (ce este în contrast cu SOAP) este că o operat,iune asupra unei resurse este bazată pe un verb HTTP folosit în combinat,ie cu un URI.

HTTP are not,iunea de verbe. Cele mai folosite sunt GET s,i POST, dar pe lângă acestea mai sunt câteva ce pot fi utilizate pentru alte operat,iuni. Lista completă a verbelor este: OPTIONS, GET, HEAD, POST, PUT,

PATCH, DELETE, TRACE s,i CONNECT.

Acestea pot fi folosite cu sensul lor semantic, astfel încât atunci când se cites,te o resursă, se poate folosi metoda GET, iar atunci când se s,terge o resursă, se poate folosi DELETE, s,i as,a mai departe.[19]



44 *CAPITOLUL 6. ASP.NET WEB API*

Figura 4

După cum se arată în Figura 4, folosind URI-ul s,i verbul corect, o resursă poate fi manipulată folosind operat,iile CRUD (Create, Read, Update, Delete). Atunci când se efectuează o cerere către server, serverul o analizează s,i construies,te răspunsul pentru a returna datele sau rezultatul clientului. Fiecare răspuns este reprezentat de o stare s,i un status HTTP, ce ar trebui folosit pentru a informa clientul cu privire la rezultatul cererii.[19]

Există cinci tipuri de statusuri HTTP:

Informational (1xx) Succes (2xx)

Redirection (3xx) Client errors (4xx) Server errors (5xx)

Fiecare tip de status are propriile detalii. De exemplu, în cazul în care cererea este efectuată cu succes, statusul răspunsului este „200 OK", după o cerere GET, dar este „201 CREATED" după o cerere POST. În cazul unui client care nu este autorizat să efectueze o cerere, statusul „403 Forbidden" ar trebui utilizat; dacă o resursă nu poate fi găsită, statusul „404 Not found" este utilizat.

1. GET

GET este folosit pentru a citi o resursă. URI-ul specifică unde se găses,te resursa pe care o citim, s,i se poate folosi header-ul Accept pentru a returna resursa într-un format specific. De exemplu cererea din Figura 5 instruies,te serverul să returneze cont,inutul în format JSON.



Figura 5

|  |  |
| --- | --- |
| *6.7. CODE ON DEMAND* | 45 |

Cererea din Figura 6 instruies,te serverul să returneze o resursă Post cu identificatorul 42, în format XML.



Figura 6

O cerere GET este considerată una sigură, as,adar aceasta nu ar trebui să modifice niciodată starea unei resurse. Serverul răspunde de obicei unei cereri GET cu statusul HTTP 200 OK dacă totul decurge bine, 404 Not found dacă URI-ul pointează către o resură inexistentă, sau 400 Bad request dacă cererea nu este corectă.[19]

1. POST

Când POST este folosit pentru a crea o resursă, datele resursei sunt tri-mise către server ca parte a corpului cererii. Serverul răspunde cu un status 201 CREATED dacă totul merge bine. Când este creată o nouă resursă, o bună practică este de a utiliza antetul Location în răspuns pentru a speci-fica URI-ul resursei nou create. Această bună practică aderă la principiul

HATEOAS.[19]

Notă: *HATEOAS (Hypermedia as the Engine of Application State)*

*Într-o aplicat,ie REST, clientul trebuie să s,tie cât mai put,ine informat,ii pentru a utiliza aplicat,ia. În mod ideal, singurul lucru pe care clientul trebuie să îl s,tie este URI-ul punctului de intrare. Toate celelalte URI-uri trebuie să fie furnizate de către server folosind anteturile de localizare sau alte meca-nisme (link-uri rel, de exemplu), pentru a informa clientul unde sunt restul resurselor. În acest fel clientul s,i serverul nu sunt legate s,i serverul ar putea schimba locat,ia resursei fără a strica clientul. Acest principiu este baza unui API REST bine conceput.*

1. PUT

PUT este folosit pentru a modifica o resursă. URI-ul specifică resursa care va fi modificată iar corpul cererii cont,ine noile valori ale resursei. Răspunsul ar trebui să cont,ină statusul 200 OK sau 204 No content în cazul în care răspunsul nu cont,ine resursa modificată. Nu este necesar să se returneze

46 *CAPITOLUL 6. ASP.NET WEB API*

URI-ul resursei în antetul Location deoarece clientul s,tie deja acest URI. [19]

PUT trebuie să fie *idempotent*, ceea ce înseamnă că rezultatul unei cereri efectuate cu succes nu depinde de numărul de execut,ii a cererii. Trebuie să fie posibil să se efectueze două apeluri identice către server, iar serverul nu ar trebui să returneze erori; al doilea apel, pur s,i simplu actualizează resursa din nou, chiar dacă aceasta nu se schimbă.

1. DELETE

DELETE este folosit pentru a s,terge o resursă. Rezultatul poate fi 200 OK sau 204 NO CONTENT dacă răspunsul nu cont,ine un body. Poate fi 404 Not found dacă URI-ul nu este corect s,i resursa nu poate fi găsită.[19]

1. Proiect Web API

Template-ul Web API este parte a template-ului de proiect ASP.NET MVC 5. Acesta este instalat implicit in Visual Studio 2012 s,i Visual Studio 2013. Pentru versiuni mai vechi de Visual Studio aceste template-uri tre-buie instalate. Structura unei aplicat,ii ASP.NET MVC 5 este prezentată în

Figura 7.

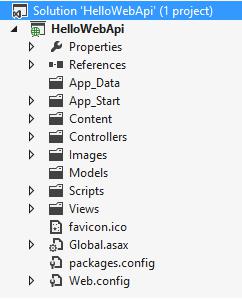


Figura 7

|  |  |
| --- | --- |
| *6.8. PROIECT WEB API* | 47 |

Cele mai importante lucruri de evident,iat sunt:

Directoarele Controllers, Models s,i Views sunt împrumutate din ASP.NET MVC. Web API foloses,te acelas,i s,ablon ca s,i MVC. Totus,i, directorul Views nu este foarte folositor în contextul Web API, des,i este posibil să se returneze un view către un client

Pe lângă directorul Views, mai sunt directoarele Images, Scripts s,i Content. Acestea nu sunt folosite de obicei, din moment ce un API este în general utilizat pentru a returna date, nu interfet,e utilizator

Directorul App\_Start este folosit pentru a configura API-ul. Acesta cont,ine diverse configurat,ii pentru a seta comportamentul API-ului

Atunci când este creat, un proiect ASP.NET Web API cont,ine un con-troller implicit(Figura 8).

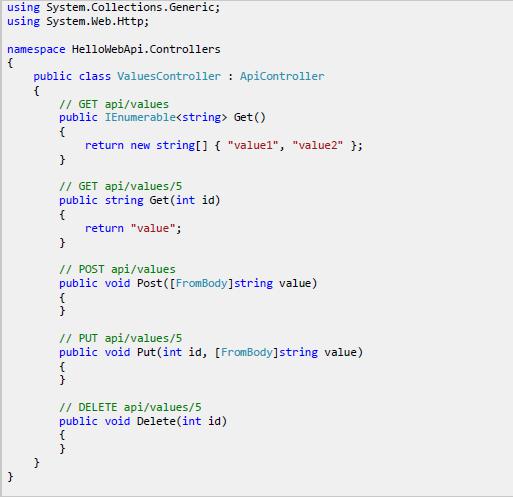


Figura 8

După cum se poate vedea în Figura 4, după instruct,iunile „using" s,i „namespace" se declară o nouă clasă, ValuesController. Această clasă mos,tenes,te din ApiController. Acest lucru este important deoarece, Api-Controller nu este „rudă" cu clasa de bază a controller-lor din ASP.NET MVC, chiar dacă au foarte multe similarităt,i. As,adar ApiController deserves,te

48 *CAPITOLUL 6. ASP.NET WEB API*

ca s,i clasă de bază pentru toate resursele ce vor fi expuse prin intermediul API-ului.

În interiorul clasei se pot observa toate verbele implicite utilizate în mani-pularea resurselor: GET, POST, PUT, DELETE. Numele metodelor din interiorul clasei este foarte important, din moment ce runtime-ul ASP.NET Web API foloses,te convent,iile ca mecanism pentru a apela metoda corectă, pe care o cerere a formulat-o. As,adar cele două metode *Get* sunt folosite pentru a returna o colect,ie de valori s,i pentru a returna o valoare cu un ID specific. Metodele *Post* s,i *Put* sunt folosite pentru a insera s,i modifica o resursă, în timp ce metoda *Delete* este folosită pentru a s,terge o resursă cu un id specific.

Ca s,i o aplicat,ie web ASP.NET MVC, proiectele Web API folosesc un sis-tem de rutare. Configurarea rutelor se efectuează într-un fis,ier denumit We-bApiConfig.cs, în directorul App\_Start. Figura 9 ilustrează cont,inutul acestui fis,ier.[19]

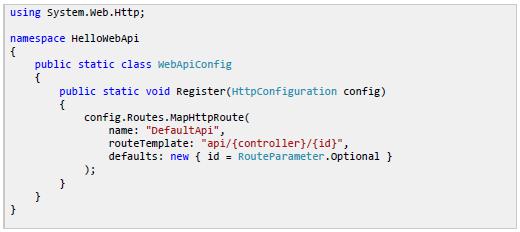


Figura 9

|  |  |
| --- | --- |
| *6.9. CICLUL DE VIAT,Ă AL UNEI CERERI* | 49 |

Această clasă cont,ine o metodă ce este invocată din clasa WebApiA-pplication, din global.asax. Această metodă înregistrează rutele folosite de aplicat,ie. Implicit, clasa ValuesController definită în Figura 8, răspunde la URI-ul */api/Values*. Este de evident,iat faptul că des,i aceste rute sunt simi-lare rutelor din ASP.NET MVC, sunt pe o stivă complet diferită. În cazul Web.API tipul rutei este IHttpRoute iar implementarea este cont,inută în assembly-ul System.Web.Http, care este un assembly complet nou, ce nu are nicio legătură cu System.Web. Fiecare rută are un nume s,i un template ce cont,ine anumit,i tokeni pentru a se potrivi cu s,abloanele de intrare.[19]

1. Ciclul de viat,ă al unei cereri

Atunci când un client trimite o cerere către o aplicat,ie ASP.NET Web Api, cererea va trece prin trei straturi, pentru a fi procesată. Componentele principale ce au un rol activ în această procesare, sunt listate în Figura 10.

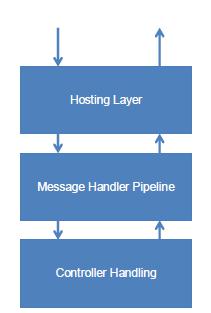


Figura 10

Următorele straturi sunt folosite atunci când o cerere este procesată:

Hosting Layer

Message Handler Pipeline Controller Handling

50 *CAPITOLUL 6. ASP.NET WEB API*

1. Hosting Layer

Primul strat este stratul de hosting, care primes,te cererea HTTP direct de la client. Stratul de hosting ar putea fi un server clasic de IIS(Internet Information Server) ce foloses,te mecanismul ASP.NET, sau o aplicat,ie self-hosted.

Rolul stratului de hosting este de a primi cereri s,i de a le converti în instant,e de HttpRequestMessage, o clasă ce reprezintă cererea. Mesajul cererii este transmis mai jos către Message Handler Pipeline. Cum este contruită această cerere depinde de tipul de hosting.

1. Message Handler Pipeline

Message Handler Pipeline reprezintă stratul de mijloc al arhitecturii pre-zentate. Acesta constă în înlant,uirea de handlere ce pot fi folosite pentru a satisface nevoile aplicat,iei. Fiecare handler este o instant,ă a clasei derivată din HttpMessageHandler ce are o metodă SendAsync, care primes,te o

instant,ă a HttpRequestMessage s,i returnează un HttpResponseMessage.[19] Fiecare din aceste handlere are o referint,ă către un InnerHandler, ce

reprezintă următorul handler din secvent,ă ce va fi apelat. Cu această arhi-tectură, fiecare cerere poate fi pre-procesată sau post-procesată de multiple handlere ce fac diferite lucruri.

Exemple de handlere de mesaje sunt HttpRoutingDispatcher ce direct,ionează cererea pe baza unei rute s,i HttpControllerDispatcher ce trimite cererea

către controller. Aceste handlere sunt deja în secvent,ă, de vreme ce sunt în colect,ia HttpConfiguration.MessageHandlers. Altele pot fi adăugate în colect,ie în timpul configurării aplicat,iei Web API.

1. Controller Handling

Acesta este ultimul strat. Stratul de controller handling primes,te mesajul cererii de la stratul anterior s,i apelează act,iunea controller-ului trimit,ând parametrii cerut,i. Task-ul este îndeplinit de HttpControllerDispatcher, ultimul handler din secvent,ă. Acesta, cu ajutorul HttpControllerDes-criptor, obt,ine o instant,ă a clasei ce implementează IHttpInterface s,i apelează metoda ExecuteAsync a acestei instant,e. Selectarea act,iunii corecte pentru a fi executată intră în îndatoririle metodei ApiControl-ler.ExecuteAsync, care bind-uies,te parametrii, execută filtrele act,iunii (dacă sunt prezente), iar mai apoi execută act,iunea propriu-zisă.[19]

Un IActionResultConverter convertes,te rezultatul act,iunii la o instant,ă HttpResponseMessage. Mesajul cererii este trimis către client folosind aceeas,i cale ca s,i a cererii.

Capitolul 7

Azure Cloud

1. Prezentare generală a cloud computing

Un data-centru local necesită gestionarea multor detalii, cum ar fi cum-părarea s,i instalarea de hardware, gestionarea virtualizării, instalarea siste-mului de operare s,i a oricăror aplicat,ii necesare, configurarea ret,elei (inclusiv a firelor necesare pentru funct,ionare), configurarea firewall-ului dar s,i confi-gurarea stocării datelor. După ce toate acestea au fost făcute, intervine apoi responsabilitatea pentru ment,inerea sistemului pentru întreg ciclul de viat,ă. Aceste lucruri înseamnă costuri foarte mari pentru hardware dar s,i costuri pentru întret,inere.

Cloud computing vine cu o alternativă modernă la un data-centru local tradit,ional. Un furnizor public de servicii cloud este complet responsabil pentru achizit,ionarea s,i întret,inerea de hardware, s,i de obicei oferă o gamă largă de servicii de platformă ce pot fi folosite. Se pot alege serviciile s,i resursele hardware s,i software ce sunt necesare, iar partea cea mai importantă este că pentru aceste servicii s,i resurse se va plăti doar atunci când vor fi folosite.

Mediile cloud oferă de obicei un portal online, us,urând astfel viat,a utiliza-torilor să administreze calcule, spat,ii de stocare, ret,ele s,i resurse de aplicat,ii. De exemplu, un utilizator poate folosi portalul pentru a crea o configurat,ie de mas,ină virtuală (VM) specificând următoarele: dimensiunea nodului de calcul (cu referire la CPU, RAM s,i disk local), sistemul de operare, software preinstalat, configurat,ia ret,elei, sau locat,ia nodului. Utilizatorul poate apoi lansa mas,ina virtuală bazată pe aceste configurat,ii s,i în câteva minute poate accesa nodul de calcul lansat.[20]

51

52 *CAPITOLUL 7. AZURE CLOUD*

Pe lângă serviciul public de cloud descris anterior, mai există cloud-uri private s,i hibride. Într-un cloud privat, se creează un mediu de cloud în propriul data-centru s,i se poate oferi access de tip „serviciu propriu" pentru a calcula resursele utilizatorilor dintr-o organizat,ie. Acest mecanism oferă o simulare a unui cloud public utilizatorilor, dar administratorul cloud-ului va fi complet responsabil de achizit,ionarea s,i întret,inerea serviciilor hardware s,i software oferite. Un cloud hibrid integrează cloud-uri publice s,i private, ofe-rind posibilitatea de a găzdui volume de muncă în cele mai potrivite locat,ii. De exemplu, se poate găzdui un website foarte mare în cloud-ul public s,i se poate asocia la o bază de dată securizată găzduită în cloud-ul privat. Micro-soft oferă suport pentru cloud-uri publice, private s,i hibride. Microsot Azure este un cloud public.

1. Comparat,ie între local s,i Azure

Cu o infrastructură locală, administratorul det,ine controlul total asupra hardware-ului s,i a software-ului ce se instalează. Istoric, aces,ti factori au dus la decizii de cumpărare a hardware-ului orientate către scalare în sus (scaling-up); s,i anume achizit,ionarea de servere cu mai multe nuclee pentru a satisface nevoile de performant,ă. Cu Azure, se pot instala aplicat,ii doar pe hardware-ul oferit de Microsoft. Acest lucru înseamnă o scalare către exterior(scale-out) a numărului de nuclee necesare pentru o aplicat,ie. Des,i există consecint,e asupra design-ului unei arhitecturi software potrivite, există dovezi clare că scalarea către exterior a hardware-ului este semnificativ mai eficientă din punct de vedere al costurilor decît scalarea în sus.

Microsoft a instalat data-centre Azure în 19 regiuni de pe glob, de la Melbourne până în Amsterdam sau din Sao Paulo până în Singapore. În plus, Microsoft are o înt,elegere cu Via21Net, făcând astfel Azure disponibil în două regiuni din China. Azure oferă flexibilitatea de a seta foarte rapid un mediu de dezvoltare s,i de testare. Acestea pot fi setate folosind script-uri, oferind astfel posibilitatea de a seta un mediu de dezvoltare sau testare, de a testa, iar mai apoi de a reveni la starea init,ială. Acest mecanism păstrează costurile foarte mici, iar mentenant,a este aproape inexistentă.

Un alt avantaj al Azure este că se pot încerca oricâte versiuni de software fără a fi nevoie de a actualiza echipamentul folosit. De exemplu, se pot verifica ramificat,iile rulării unei aplicat,ii folosind Microsoft SQL Server 2014 în locul Microsoft SQL Server 2012, se poate crea o instant,ă de SQL Server 2014 s,i se poate rula o copie a unor servicii pe noua bază de date, toate fără a fi nevoie să se actualizeze hardware-ul sau să se facă noi conexiuni (fizice, cu fire).[20]

|  |  |
| --- | --- |
| *7.3. CE OFERĂ CLOUD* | 53 |

1. Ce oferă Cloud

Cloud computing este de obicei clasificat în trei categorii: Saas, Paas s,i IaaS. Totus,i, pe măsură ce cloud cres,te, diferent,ele între aceste trei categorii încep să dispară.

1. SaaS: Software as a Service

SaaS este software ce este găzduit central s,i întret,inut pentru consu-matorul final. De obicei este bazat pe o arhitectură pe mai multe stra-turi – o singură versiune a aplicat,iei este folosită pentru tot,i consumatorii. Poate fi scalată către exterior la multiple instant,e pentru a asigura cea mai bună performant,ă în toate locat,iile. Software-ul SaaS este de obicei licent,iat printr-o taxă lunară sau anuală.

Oﬃce 365 este un model prototip al SaaS. Subscriberii plătesc o taxă lunară sau anuală, s,i primesc Exchange As a Service(Outlook online s,i local), Storage as a Service (OneDrive), s,i restul suitei Microsoft Oﬃce. Subscriberii primesc de obicei ultimile versiuni de software. Alte exemple de Saas includ Microsoft One Drive, Dropbox, WordPress sau Amazon Kindle.[20]

1. PaaS: Platform as a Service

Cu PaaS, se pot instala aplicat,ii în mediul de hosting oferit de către distribuitorul de servicii cloud. Programatorul dezvoltă aplicat,ia iar distri-buitorul de servicii cloud asigură abilitatea de a instala s,i rula aplicat,ia. Acest mecanism us,urează viat,a programatorilor, s,i îi scutes,te de administra-rea infrastructurii, permit,îndu-le să se focuseze strict pe dezvoltare.

Azure oferă mai multe PaaS-uri, inclusiv Azure Websites sau Azure Cloud Services. As,adar dezvoltatorii au multiple variante de a instala o aplicat,ie fără să s,tie nimic despre infrastructură. Aces,tia nu trebuie să creeze mas,ini virtuale, nu trebuie să folosească Remote Desktop pentru a se loga pe acestea.

Scalarea spre exterior a unui serviciu Azure de calcul se rezumă de obicei la cres,terea numărului de instant,e , moment în care Azure instalează o nouă mas,ină virtuală s,i software-ul. Azure se ocupă chiar s,i de mecanismul de load-balancing(echilibrare). Pentru a instala o nouă versiune, o aplicat,ie trebuie doar să fie republicată.[20]

54 *CAPITOLUL 7. AZURE CLOUD*

1. IaaS: Infrastructure as a Service

Un distribuitor de servicii IaaS rulează s,i întret,ine servere rulând software de virtualizare, permit,ând astfel crearea de mas,ini virtuale ce rulează pe infrastructura distribuitorului de servicii cloud. În funct,ie de distribuitor, se pot crea mas,ini virtuale ce rulează Windows sau Linux s,i se poate instala orice pe acestea. Azure oferă posibilitatea de a configura ret,ele virtuale, load balancers, mecanisme de stocare, s,i de a folosi multiple servicii. Folosind IaaS, singurele lucruri ce nu se pot controla sunt hardware-ul s,i software-ul de virtualizare.

Azure Virtual Machines este oferta Azure ca s,i IaaS, s,i este o alegere populară atunci când se migrează servicii către Azure deoarece IaaS permite utilizarea modelului de interschimbare pentru migrat,ii. Se pot configura mas,ini virtuale similare infrastructurii pe care rulează în prezent o aplicat,ie s,i se poate migra aplicat,ia foarte us,or.[20]

1. Servicii Azure

Azure include multiple servicii, dar cele mai importante sunt:[20]

Servicii de calcul: Microsoft Azure Cloud Services, Azure Virtual Ma-chines, Azure Websites s,i Azure Mobile Services

Servicii de date: Microsoft Azure Storage, Azure SQL Database s,i Redis Cache

Servicii de aplicat,ii: Servicii ce pot fi folosite pentru a construi s,i opera aplicat,ii, cum ar fi Azure Active Directory, Service Bus pentru conec-tarea de sisteme distribuite

HDInsight pentru procesarea de date cu volum mare, Azure Scheduler s,i Azure Media Services

Servicii de ret,ea: Caracteristici Azure cum ar fi Virtual Networks, Azure Content Delivery Network s,i Azure Traﬃc Manager

Capitolul 8

Arhitectura aplicat,iei

Aplicat,ia este formată din trei module importante. Primul modul este reprezentat de aplicat,ia client - aplicat,ie Android, al doilea modul este re-prezentat de server - aplicatie web s,i aplicat,ia care se ocupă cu preluarea s,i prelucrarea datelor de pe Internet s,i depunerea acestora în baza de date aflată în cloud - aplicat,ie Java Spring.

1. Aplicat,ia client

Aplicat,ia client este o aplicat,ie pentru dispozitive mobile care au sistem de operare Android. Aceasta a fost implementată cu ajutorul limbajului Java utilizând ca s,i mediu de dezvoltare Android Studio, astfel că am folosit o serie de instrumente interactive care se îmbină în armonie pentru a crea o interfat,ă prietenoasă, us,or de utilizat de către det,inătorii de smartphone-uri Android. Aplicat,ia a fost dezvoltată având ca s,i nivel API minim versiune 14, acoperind astfel 87,9% din toate dispozitivele active în Google Play Store, începând cu versiunea de Android Ice Cream Sandwich. Am ales această versiune de API datorită faptului că permite utilizarea de funct,ionalităt,i noi care în versiunire anterioare nu sunt disponibile, rezultând astfel un design mai modern s,i mai intuitiv. Mai mult, numărul de device-uri active cu un nivel API inferior este în scădere, de asemenea s,i numărul de aplicat,ii dezvol-tate pentru versiunile inferioare a scăzut semnificativ în ultima perioadă.

1. Aplicat,ia desktop Java

Acest modul se ocupă cu preluarea s,i parsarea informatiilor despre pro-duse cu ajutorul API-ului Jsoup HTML Parser1.

1vezi capitolul 5 - Jsoup API

55

56 *CAPITOLUL 8. ARHITECTURA APLICAT,IEI*

Această aplicat,ie se ocupă preluarea produselor de pe anumite site-uri s,i în funct,ie de fiecare categorie va rezulta o listă de produse care apoi va fi filtrată s,i în final alegându-se doar produsele pentru care se aplică redu-cere, pentru a nu stoca informat,ii pe care nu le vom folosi ulterior. Atfel după preluarea datelor de pe Internet se asigură persistent,a acestora prin intermediul serviciului REST.

1. Aplicat,ia server

Aplicat,ia server este reprezentată de un serviciu REST care a fost im-plementat cu ajutorul templatelul Web API regăsit în ASP.NET2 Acesta reprezintă legătura între cele două module prezentate anterior întrucât prin intermediul serviciului se depun datele în baza de date, se preiau datele din baza de date s,i se modifică cu ajutorul operat,iilor REST. O interfat,ă, ce defines,te contractul între server s,i viitorii client,i, cont,ine toate metodele la care client,ii vor avea acces. Aceste metode sunt implementate în clasa servi-ciului. Persistenta datelor este realizata într-o bază de date aflată în cloud3. Comunicarea dintre baza de date s,i server este efectuată folosind ORM-ul (Object Relationing Model) Entity Framework. Acest model este util deoa-rece întreaga bază de date este cartografiată în server. Acest lucru înseamnă că pentru fiecare tabel din baza de date, este creată o clasă iar clasele vor fi puternic conectate intre ele. De asemenea orice modificare adusă bazei de date este us,or de integrat s,i în server.

2vezi capitolul 6 - REST

3pentru detalii despre cloud computing vezi Capitolul 7

Capitolul 9

Capitolul 8

1. PromON

Log In/Sign Up

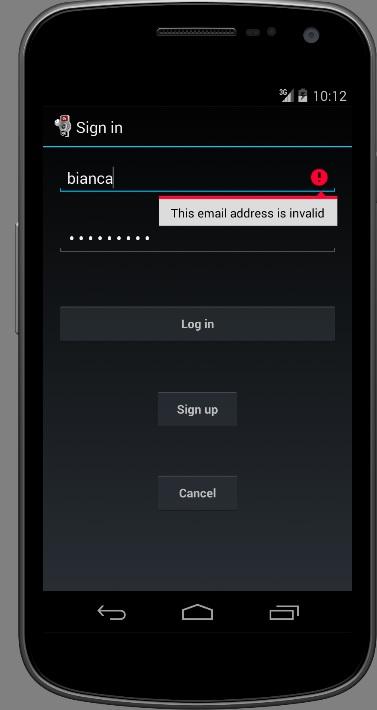
Meniul de start al aplicat,iei client este reprezentat de fereastra de Log In/Sign Up care este afis,ată doar în cazul în care utilizatorul nu este deja autentificat. Aceasta cuprinde două text box-uri în care clientul îs,i va intro-duce un e-mail s,i o parolă necesare pentru autentificare sau pentru crearea unui cont nou, caz în care va fi necesară o confirmare a parolei.



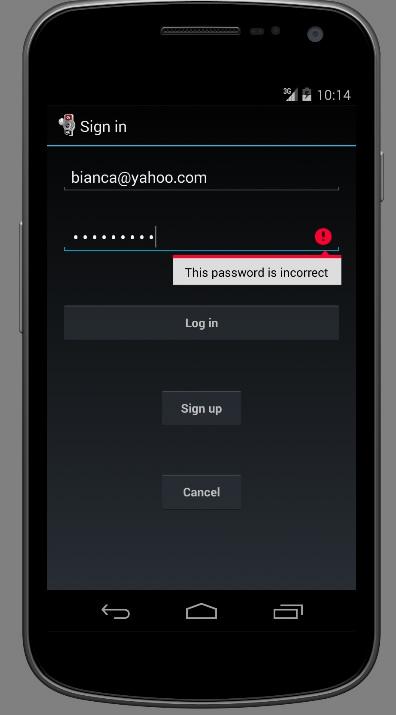
57

58 *CAPITOLUL 9. CAPITOLUL 8*

Login



Atent,ionare e-mail invalid



|  |  |
| --- | --- |
| *9.1. PROMON* | 59 |

Atent,ionare parolă invalidă

Pagina principală

După conectare se va deschide o pagină în care, vor fi afis,ate primele 10 produse sortate în funct,ie de cel mai mare discount acordat. Acestea se vor schimba dupa 5 secunde timp în care dacă utilizatorul este interesat va putea apăsa pe butonul „cumpără” care îl va redirect,iona spre site-ul de unde poate achizit,iona produsul.

Meniul slide-in

Pentru a oferi un design elegant s,i nu foarte încărcat, pentru a naviga prin celelalte funct,ionalităt,i ale aplicat,iei am ales să utilizez un meniu slider care va sta ascuns pe parcursul act,iunilor de pe o anumită pagină dar va putea fi oricând vizibil printr-o simplă atingere a ecranului dinspre stânga spre dreapta sau prin atingerea iconit,ei de meniu. Astfel toate modulele aplicat,iei sunt grupate s,i pot fi accesate rapid s,i simplu, comunicarea între acestea fiind de asemenea printr-o simpla atingere a elementului din meniu corespunzător paginii care se dores,te a fi deschisă. Unul dintre motivele pentru care am ales să dezvolt aplicat,ia pentru versiunile de Android pornind de la IceCreamSandwich este utilizarea acestui tip de meniu care din punctul meu de vedere are un impact mare fiind un instrument modern s,i din ce în ce mai utilizat în aplicat,ii importante precum Facebook, YahooMail, GMail etc. Asfel, utilizatorii Android sunt obis,nuit,i cu utilizarea acestui navigator s,i pot folosi aplicat,ia cu us,urint,ă.

Categorii

Acest fragment al aplicat,iei cuprinde o listă de categorii de produse dispo-nibile, personalizate cu o iconit,ă sugestivă pentru impact vizual s,i un buton comutator prin care un client îs,i poate alege metoda de afis,are a categoriilor, sortate în funct,ie de nume, alfabetic, sau în functie de numărul de accesări, cele mai populare categorii fiind listate primele în acest caz.

60 *CAPITOLUL 9. CAPITOLUL 8*



Listă categorii sortate alfabetic



Listă categorii sortate în funct,ie de numărul de accesări

|  |  |
| --- | --- |
| *9.1. PROMON* | 61 |

Produse

După ce utilizatorul selectează o categorie din listă, se deschide o nouă fereastră, ce cuprinde o listă cu produsele care apartin categoriei alese. De asemenea acestea pot fi sortate, utilizând un buton comutator, alfabetic sau descrescător după valoarea discountului care se oferă pentru fiecare produs în parte. După ce alege produsul dorit, pentru a fi vizionat în detaliu se deschide o nouă fereastră în care se pot vizualiza pret,ul curent, pret,ul anterior, detalii despre produs s,i o poză semnificativă. Această pagină corespunde cu pagina principală, în care sunt afis,ate însă doar 10 produse, considerate produsele de top. De asemenea, pentru a cumpăra un produs, apare un buton, care , o dată apăsat va deschide browserul principal al telefonului s,i va deschide pagina corespunzătoare produsului curent.

Setări

Meniul de setări cuprinde opt,iunea de a alege o temă pentru aplicat,ie prin selectarea acesteia dintr-un meniu drop-down. O dată selectată o temă, clientul trebuie să apese butonul Aplică pentru ca tema curentă să se modifice cu cea aleasă. Acest fragment de aplicat,ie este oferă posibilitatea utilizatorul de a-si personaliza aplicat,ia, este util însă s,i pentru sănătatea ochilor întrucât în funct,ie de lumină este recomandat să alegem culori care să nu ne afecteze vederea.



Pagină setări

62 *CAPITOLUL 9. CAPITOLUL 8*

Termeni s,i condit,ii

Orice aplicat,ie cuprinde o serie de termeni s,i condit,ii, aces,tia au fost acceptat,i o dată cu instalarea aplicat,iei, însă este de preferat să fie disponibili pentru o recitire ulterioară s,i în cazul în care utilizatorul nu este de acord să poată dezinstala aplicat,ia.

Log Out

Acest fragment se referă la ies,irea din cont s,i o dată apăsat butonul de Log Out, utilizatorul va fi redirect,ionat către pagina de Log In, unde va putea să se autentifice cu un alt cont deja existent, să-s,i creeze un cont nou sau pur s,i simplu să iasă din aplicat,ie.



Pagină log out

Bibliografie

1. Onur Cinar, 2015, *Android Quick APIs Reference*, Editura APRESS.
2. Dave Smith, 2015, *Android Recipes: A Problem-Solution Approach for* *Android 5.0*,Editura APRESS.
3. Wei-Meng Lee, 2012 *BEGINNING Android 4 Application Development*, Editura Wrox Press
4. Sayed Y. Hashimi and Satya Komatineni, 2009, *Pro Android*, Editura

APRESS.

1. Open Handset Alliance, 2007, Industry Leaders Announce Open Plat-form for Mobile Devices. Press release., http://www.openhandsetalliance.com/press\_110507.html

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [6] Google’s | Android | parts | ways | with | Java | in- |
| dustry |  |  | group,http://www.cnet.com/news/ | | | |

googles-android-parts-ways-with-java-industry-group/

1. General Android 2008, http://code.google.com/android/kb/general.html
2. Native C application for Android, http://benno.id.au/blog/2007/11/13/android-native-apps
3. Android version history,

http://en.wikipedia.org/wiki/Android\_version\_history# Pre-commercial\_release\_versions

1. http://www.oracle.com/technetwork/java/
2. Barry Burd PhD, 2014, Beginning Programming with Java For Dum-mies, 4th Edition, Wiley eBooks
3. Ştefan Tanasa, Cristian Olaru, Stefan Andrei, "Java de la 0 la expert", Polirom, 2003.
4. Yakov Fain, 2011, Java Programming 24-Hour Trainer, Wrox eBooks

63

64 *BIBLIOGRAFIE*

1. Yvor Horton, 2011, Beginning Java , Java 7 Edition, John Wiley & Sons, Inc.
2. Adrian Deaconu, 2007, Programarea în C/C++ s,i aplicat,ii , Grupul microInformatica, Cluj-Napoca
3. Document Object Model (DOM). http://www.w3.org/: W3C., 2012
4. Lawrence Rosen, OPEN SOURCE LICENSING, p.85 (Prentice Hall PTR, 1st ed. 2004)
5. jsoup: Java HTML Parser, https://jsoup.org/
6. Emanuele Delbono, 2013, ASP.NET Web API Succinctly, Syncfusion.
7. Michael Collier, Robin Shahan, 2015, Fundamentals of Azure, Microsoft Press. Ed. APRESS.