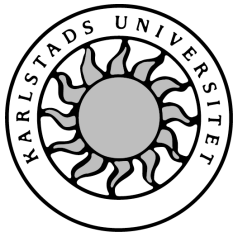


Datavetenskap

Tomas Green och Johan Käll

Neverlost Calendar - En Applikation för iPhone och Android

Kandidatuppsats

2010-05-27

Neverlost Calendar - En Applikation för iPhone och Android

Tomas Green och Johan Käll

Denna uppsats är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Tomas Green och Johan Käll

Godkänd, 2010-06-09

Opponent: Christoffer Karlsson och Jonas Östlund

Handledare: Per Hurtig

Examinator: Martin Blom

Sammanfattning

Denna uppsats beskriver vårt arbete med att utveckla en schemaapplikation mot plattformarna iPhone och Android. Syftet med vårt arbete var att förenkla vardagen för studenter vid Karlstad universitet genom att sammanfoga universitetets schemasystem Neverlost med kalendertjänsten Google Calendar. Utförandet av denna uppgift utgick från en design men resulterade i två olika implementationer, en för respektive plattform. I denna uppsats kommer plattformarna och dess historia samt utförandet och resultatet av vårt arbete att beskrivas i detalj.

Abstract

This thesis describes our work creating an schedule application for the iPhone and Android platforms. The purpose of our work was to make everyday tasks easier for the students at Karlstad university by joining the university schedule application Neverlost with the calendar service Google Calendar. The implementation of this task was derived from one design but resulted in two different implementations, one for each platform. During this thesis we will describe the platforms and their history along with the implementation and result of our design.

Tack

Vi vill tacka vår handledare Per Hurtig för ett utmärkt arbete i att vägleda oss under utförandet av denna uppsats.

Innehåll

Figurer

Kapitel 1

Inledning

Mobiltelefonanvändandet har förändrats enormt de senaste åren. Det handlar inte längre om att ha den minsta telefonen eller möjligheten att byta ringsignal. Breddandet av internet har gett oss ett allt större behov av att hela tiden ha tillgång till stora mängder information som kartor, live-tv, online-spel och sociala medier. Detta innebär också att kraven på kortare responstider och avancerade gränssnitt blir allt viktigare. I takt med att försöka tillgodose dessa krav har nu mobiltelefonerna alltmer börjat likna datorer. Figur 1.1 ser vi två exempel på så kallade smarthpones, till vänster ser vi Apple iPhone 3GS och till höger den Androidutrustade Google Nexus One.

1.1

Figur 1.1: Apple iPhone 3GS och Google Nexus One.

De höga kraven på tillgång till information tillsammans med att elektroniska komponenter blivit allt billigare och att uppkopplingshastigheten ökar har gjort att marknaden för de mer avancerade mobiltelefonerna exploderat. Även om mobiltelefonerna nu är kraftfulla kan de inte ersätta en dator och därmed inte köra stora och komplicerade program. Men med den växande popularitet av enheterna har en helt ny marknad dagats, en marknad av små och kompakta applikationer specifikt utvecklade för dessa enheter. Distributionen av dessa applikationer är i stor utsträckning centralstyrd. Vanligtvis köper och laddar man hem applikationer från enstaka webbplatser men nu ligger distributionen istället på digitala applikationstorg kopplade till kreditkortstjänster. Tillgängligheten och mognaden av utvecklingsverktyg och hjälptjänster har gett oss möjligheten till stor variation av applikationer. Allt från spel och organisatoriska verktyg till karttjänster med GPS.

Figur 1.2: Applikationen Neverlost Calendar implementerad på iPhone 3GS och Google Andorid Nexus One.

Vårt examensarbete beskriver utvecklingen av en applikation framtagen för plattformarna iPhone och Android. Applikationens syfte är att förenkla vardagen hos studenterna vid Karlstads universitet (KAU) genom utöka tillgängligheten av kursscheman. I dagsläget finns alla scheman tillgängliga som webbdokument via en länk till schemaläggningssystemet

Neverlost på tillhörande kurs webbplats. Det här systemet är enligt oss föråldrat och saknar både flexibilitet och tillgänglighet. Vi ser det som en stor fördel om man kan hantera och manipulera schemats innehåll utifrån studentens behov och önskemål. Därför har vi konstruerat en applikation som kopplar samman Neverlost och kalendertjänsten Google Calendar. Denna kalender använder sedan mobiltelefonerna och ger oss ständig tillgång till våra scheman med möjligheter att redigera och exempelvis lägga till påminnelser. Figur 1.2 visar bilder av applikationen implementerad till iPhone och Android.

1.1 Disposition

Resten av uppsatsen är uppdelad i 7 kapitel: Bakgrund, Neverlost Calendar, Design, Användargränssnitt, Implementation och Avslutande kommentarer.

I det första kapitlet kommer vi att beskriva plattformarna som utvecklingen av applikationen skett mot, dess historia och kringliggande programvara. Bland detta ingår bakgrunden till iPhone och Android samt de programmeringsspråk, utvecklingsverktyg och ramverk vi använt i implementationen av vår applikation.

Det andra kapitlet, ”Neverlost Calendar” beskriver vad vi har gjort, varför vi gjorde det och överskådligt vilka komponenter som krävdes. Kapitlet kommer beröra schemasystemet Neverlost, kalender-tjänsten Google Calendar samt Google Data API.

I kapitlet ”Design” kommer vi att beskriva hur de olika implementationerna representeras med grafiska användardgränssnitt och hur användaren interagerar med telefonerna. I kapitlet förklaras även vilka riktlinjer som legat till grund för designen av GUIt, samt skillnaderna mellan plattformarna iPhone och Android.

Nästa kapitel, ”Implementation” beskriver hur applikationen är implementerad för plattformarna iPhone och Andorid. Kapitlet kommer även beskriva vad designmönstret Model View Controller är och hur det är tillämpat för våra ändamål men dessutom beskriva vissa problem med tolkningen den HMTL-kod som Neverlost genererar.

Slutligen, i kapitlet ”Avslutande kommentarer” kommer vi bland annat utvärdera utvecklingsarbetet och de verktyg vi använt i framtagandet av vår applikation. Detta innefattar även Google Data API och de certifikat vi behövt för att testa applikationen på våra mobiltelefoner. Vi kommer även tala om de visioner vi har samt och de förbättringsåtgärder vi kan applicera på applikationen och slutligen kort gå igenom vad vi tyckte om projektet i sin helhet och vad vi lärt oss under tiden.

Kapitel 2

Bakgrund

Det här kapitlet kommer beskriva plattformarna som utvecklingen av applikationen skett mot, dess historia och kringliggande programvara. Bland detta ingår bakgrunden till iPhone och Android samt de programmeringsspråk, utvecklingsverktyg och ramverk vi använt i implementationen av vår applikation.

2.1 Apple iPhone

Det här avsnittet ger en överblick av iPhone och dess historia, samt om hur företaget Apple och deras produkter såsom iPod har legat till grund för utvecklingen av iPhone. Avsnittet beskriver även de kringliggande tjänsterna App Store och iTunes Store samt operativsystemen Mac OS X och iPhone OS.

2.1.1 Apple Inc.

Apple Inc. är ett Amerikanskt multinationellt företag som grundades 1976 och som är inriktat på tillverkning och design av elektroniska produkter samt utvecklingen av mjukvara [30]. Under åren har Apple varit ett av de största företagen inom branschen med konkurrenter som Microsoft och IBM. De har under alla år varit starkt inriktade på GUIs (Graphical

User Interface) och var det första företaget i branschen som släppte persondatorer med GUI.

Utöver persondatorer, stationära som bärbara, har de dessutom utvecklat och tillverkat en rad andra produkter såsom nätverksrouters och digitala TV-boxar. Sedan 2001 har de även etablerat sig på teknikmarknaden som den största försäljaren av portabla musikspelare med produkten iPod. I och med den växande popularitеten av så kallade smartphones har Apple sedan 2008 dessutom utvecklat tagit sig in mobiltelefonmarknaden med produkten iPhone.

2.1.2 Mjukvara

Sedan 2002 har Apple arbetat med operativsystemet Mac OS X som till en början utvecklades av företaget NeXT Inc. NeXT Inc. startades av Steve Jobs efter att han tillfälligt tvingades lämna VD-posten på Apple. NeXT Inc. skapade tillsammans med Sun Microsystems det POSIX-baserade och objektorienterade operativsystem NeXTStep. NeXTStep låg sedan till grund för Mac OS X varav iPhone OS är en avskalad och för hårdvaran optimerad version av Mac OS X.

Den mest använda applikationen som Apple har utvecklat är iTunes. iTunes skapades i samband med lanseringen av musikspelaren iPod och användas för att organisera och spela upp mediafiler på den. Dessa filer kan i sin tur synkronisera med en iPod. Genom iTunes går det även att göra säkerhetskopior av sin enhet, uppdatera programvara, organisera applikationer och synkronisera till exempel adressbok och kalender [39].

iTunes är även kopplad till två av Apples online-tjänster, iTunes Store och App Store. Tjänsten iTunes Store är till för försäljningen av digital media såsom film och musik. I dagsläget svarar tjänsten för ca 70 procent av marknaden [40].

Med App Store kan man ladda ner och köpa en mängd olika applikationer, allt från spel, organisatoriska verktyg eller tredjeparts-tjänster såsom externt lagringsutrymme. App Store finns tillgänglig både genom iTunes och som en separat applikation i iPod Touch och

iPhone. Konceptet med att sälja applikationer genom en tjänst har visat sig vara en stor succé och sedan lanseringen 2008 finns det nu över 140.000 applikationer till försäljning var av dessa har laddats ner ca tre miljarder gånger [29]. På grund av dess framgång har tjänsten blivit någon av en trend som anmanats av andra bolag, inte minst Google med motsvarigheten Market.

2.1.3 Mobila enheter

Apple har som tidigare nämnts släppt ett antal mobila enheter sedan 2001. Den allra första som slog igenom var iPod. iPod är en portabel musikspelare som lanserades under 2001 och har sedan 2004 dominerat marknaden med över 240 miljoner sålda exemplar och i en mängd olika modeller. Den mest avancerade är iPod Touch som delar mycket av sin hårdvara och funktionalitet med iPhone [38].

2.1.4 iPhone

Efter de stora framgångarna med iPod började Apple snabbt fundera över hur denna teknik skulle kunna användas i andra enheter, då framförallt med tanke på den sedan länge populära mobiltelefonen. Motorola ROKR E1 släpptes i September 2005 och var den första mobiltelefonen kopplad till iTunes för synkronisering av musik. Den använde sig av Apple-teknik för att spela upp musik men Apple ville till en början inte att telefonen skulle konkurrera med iPod, därför sattes en max-gräns på 100 låtar. Detta resulterade i att ROKER E1 aldrig riktigt etablerade sig på marknaden.

Figur 2.1: Bild av en Apple iPhone 3GS.

Under 2007 släppte Apple sin första egna mobiltelefon nämligen iPhone. iPhone är en hybrid mellan en handdator, mobiltelefon och musikspelare (en så kallad smartphone) [36]. Den första generationens iPhone hade endast stöd för telefoni och surfning via trådlösa lokala nätverk men efter lanseringen under 2007 har det släppts två nya versioner, iPhone 3G och iPhone 3GS. iPhone 3G hade stöd för GPS, videokamera, 3G för telefoni och internetanslutning, men den hade även högre processorhastighet och större lagringsutrymme. Året efter, 2008 kom nästa version nämligen iPhone 3GS som kan ses i Figur 2.1. Den hade mer än två gånger så hög processorhastighet och lagringsutrymme samt bättre kamera.

2.2 Android

Detta avsnitt beskriver Android som operativsystem, hur det uppkom och vilka förändringar som skett fram till idag. Vi kommer även att gå igenom vad de olika versionerna innebär och vilken funktionalitet som användaren kan förvänta sig av en mobil enhet tillverkad för Android.

2.2.1 Bakgrund

Google som genom sin sökmotor blivit ett av världens mest framgångsrika företag, började under 2005 köpa upp mindre företag med inriktning på mobiltelefonbranschen. Ett av de första företag som Google köpte upp var Android Inc. med Andy Rubin i spetsen. Efter det följde en ström av bolag och nyckelpersoner som under Rubins kommando skulle ta fram ett nytt flexibelt operativsystem för mobila enheter. Detta operativsystem lanserades under 2007 under namnet Android med logon som syns i Figur 2.2. Under 2006 blir det än tydligare hur Google breddade sina tjänster mot mobiltelefonmarknaden genom att optimera och ytterligare tillgängliga sina tjänster via webbläsare i mobila enheter.

Figur 2.2: Android logo.

2.2.2 Open Handset Alliance

Den 5:e November 2007 bildade Google tillsammans med diverse mjukvaru- och hårdvarutillverkare organisationsalliansen Open Handset Alliance (OHA). Kort efter att organisationen bildats överlät Google projektet Android till OHA och visade med det tydligt att Android är ett öppet operativsystem utvecklat av medlemmar i industrin.

Android är OHAs stora projekt med öppen källkod och är licensierat under Apache v.2 [2] som lämpar sig för kommersiella ändamål. Apache v.2 kräver att upphovsrätten

bevaras samt meddelas i källkoden och ansvarsfriskrivning. Medlemmar som använder sig av Android i sina mobila enheter är inte skyldiga att sprida sina produktspecifika förändringar/tillägg av Android till OHA. Android innehåller inte några programvaror eller tjänster som är knutna till en specifik medlem i OHA. Vill en tillverkare inkludera Hotmail som är Microsofts e-postlösning istället för den största konkurrenten Google Gmail är det helt öppet att göra så. Det gör att Android blir ett attraktivt OS för tillverkare och samtidigt kan tillverkare stå redo med senaste mjukvaran till pressade priser [20].

2.2.3 Android

Första telefonen som körde Android med inriktning mot konsumenter släpptes den 22:a Oktober 2008. Telefonen var tillverkad av mobiltelefontillverkaren HTC [13] men kom ut på marknaden under namnet T-mobile G1 [5]. Fram till slutet av 2009 fanns det 18 enheter på marknaden med Android OS implementerat. Google valde att öka takten ytterligare och den 5:e Januari 2010 släpper de sin egna mobiltelefon under namnet Nexus One. Android-telefoner kan till utseende och användargränssnitt skilja sig en hel del. I Figur 2.3 kan man se skillnaderna mellan två konkurrerande Android-telefoner, Nexus One och SonyEricsson X10. Tillverkaren av Nexus One är HTC som i nära samarbete med Google under utvecklingen lagt nivån högt för vilka komponenter som ska finnas i telefonen. Google har valt att som försäljningsmetod enbart sälja telefonen via internet och ej operatörssålt.

Figur 2.3: Android-telefonerna Nexus One och SonyEricsson X10.

2.2.4 Market

Market tillhandahåller applikationer för Android-telefoner. Användare kan med internetuppkoppling ladda ner nya program eller spel till sin Android-telefon. Den 17:e Maj 2010 fanns det totalt 53958 st [1] applikationer att välja bland. Market är en liknande tjänst som Apples App Store tillhandahålls för iPhone. Market är däremot mer öppet jämfört med App Store eftersom applikationer kan läggas upp utan att de granskas före. Genom market kan användarna sätta betyg på applikationer men även anmäla dem som olämpliga och till översyn av administratörer.

2.2.5 Olika Versioner av Android

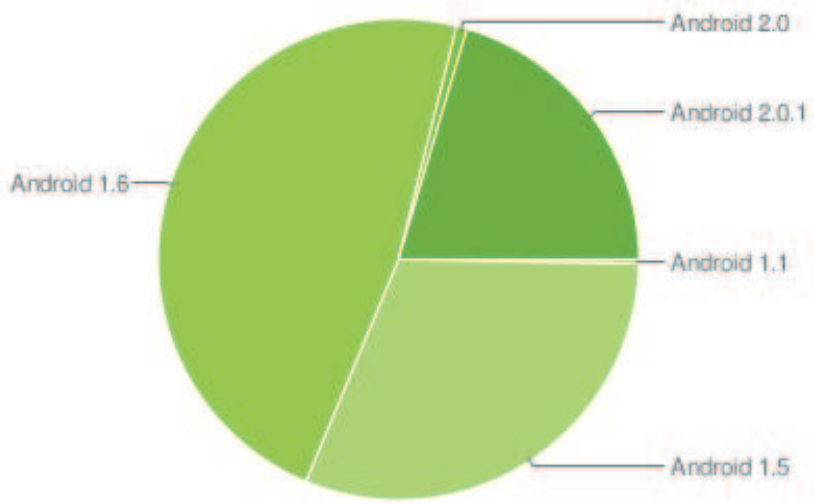
Efter den första lanseringen i September 2007 gick det fort mellan uppdateringarna, vilket kan ha att göra med att den öppna källkoden har levt sitt eget liv. Android utvecklas i den takt som anslutna medlemmar tar fram nya produkter och med nya krav. Dessa

förändringar är något som andra tillverkare i sin tur kan ta del av. Detta är ett Win-Win koncept där tillverkare kan göra det som de är bäst på, det vill säga att ta fram bra mobila enheter för konsumenterna så att de till fullo kan utnyttja den nya tekniken. På detta sätt kan medlemmarna i OHA verka och ta del av de synergieffekter som skapas av att arbeta på en gemensam och bred plattform.

En av de första implementationerna av Android som riktade sig till den stora massan konsumenter var Android Cupcake version 1.5 (släpptes 30:e April 2009). I dagsläget så är Cupcake ganska ouppdaterad. Eftersom det är upp till till tillverkarna att se till att nya versioner av Android är tillgängliga i deras mobiltelefoner hamnar visas versioner inom tid på efterkälken [17].

Den 15:e September 2009 kom Donut, version 1.6 av Android som nu hade ytterliggare funktionalitet till kameran men även med röstigenkänning, röstsökning, utökat snabbsökregister inkluderat kontakter, bokmärken och historia, allt direkt från hemskärmens sökfunktion. I Donut fick även Android Market ett ansiktslyft med bland annat stöd för att lägga upp skärmdumpar av applikationer så att användarna lättare ska kunna avgöra om applikationen är något för dem [18].

Motorola Druid var den första telefonen med stöd för Android Eclair 2.0 (26:e Oktober 2009). Droid blev väldigt populär i USA under julhandeln 2009 vilket kan styrkas av statistik över använda Android-versioner från Market (Figur 2.4). Eclair hade stora förbättringar i prestanda och ett användargränssnitt som hjälpte Android att möta upp kraven från konsumenter om ett modernare gränssnitt. Den nya versionen hade nu stöd för större skärmar vilket var ett förberedande inför den framtida marknaden av hybrider mellan mobiltelefoner och bärbara datorer. Den inbyggda webbläsaren fick dessutom en rad uppdateringar i gränssnittet samt stöd för HTML5 [28] för att utöka möjligheterna för applikationer helt knutna till webben [19].

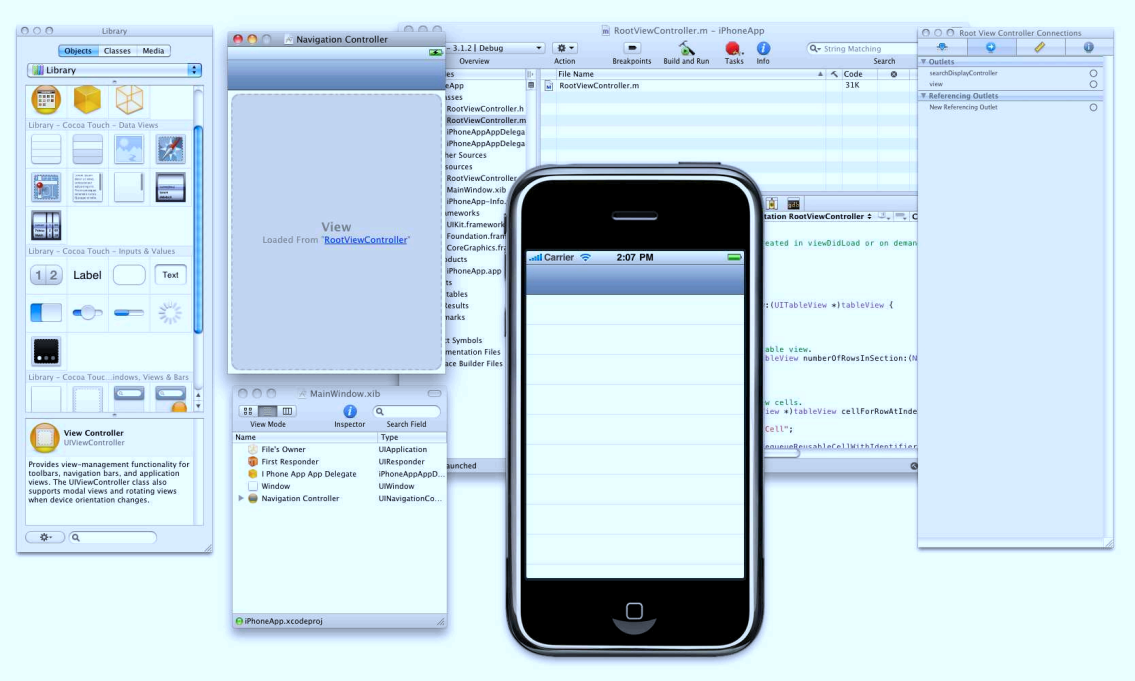


Figur 2.4: Operativsystemfördelning (versioner) över enheter som använde sig av Android Market under en 14-dagars period fram till den 4:e Jan 2010.

2.3 Utvecklingsmiljö

Det här avsnittet går igenom de verktyg och programmeringsspråk som används för att utveckla applikationer till iPhone och Android. Avsnittet om iPhone kommer att beskriva programmeringsspråket Objective-C och ramverket Cocoa samt verktyget Xcode och Interface Builder. I avsnittet om Android beskrivs i sin tur verktyget Eclipse och Android-specifika tillägg i programmeringsverktyget Eclipse så som den virtuella maskinen Dalvik och simulatorn AVD.

2.3.1 iPhone SDK



Figur 2.5: iPhone SDK 3.X, Mac OS 10.6.

Figur 2.5. är en bild av olika komponenter i den programmeringsmiljö som används för att utveckla programvara till iPhone. Programpaketet, eller det som kallas Software Development Kit (SDK) innefattar allt från olika bibliotek, kod-redigeringsverktyget XCode, Interface-builder för att skapa GUIs samt iPhone Simulator vilket är en virtuell enhet med iPhone OS installerat [37]. Precis som andra programmeringsverktyg har det givetvis stöd för versionshantering, diagram, refactoring och en mängd andra funktioner.

Det huvudsakliga programmeringsspråket som iPhone bygger på heter Objective-C[7] och är ett objektorienterat språk baserat på språket C [12]. Objective-C kan ses som ett tunt lager ovanpå C vilket innebär att det är helt möjligt att kompilera C-kod med en Objective-C kompilator eller att skriva C-kod direkt inuti en Objective-C klass. Syntaxen i Objective-C efterliknar programspråket Smalltalk där objekt anropas med hjälp av meddelanden.

Smalltalk var det första helt objektorienterade programmeringsspråket [42] och tanken bakom dess struktur var att förena maskinen med människan och hur vi ser på världen. Det har under åren influerat många av våra moderna programmeringsspråk så som Java [24], PHP [14], Ruby [15] och även Objective-C [44]. Men även om Objective-C syntaxen delar mycket med Smalltalk så skriver man den inkrementella koden precis som i C. Det finns dessutom möjlighet att använda så kallade punkt-operatorer. Språket har även stöd för vad som kallas för Objective-C++ [43] med vilket menas att man i koden kan instansiera och anropa objekt skrivna i C++.

Sammanfattningsvis så är Objective-C ett väldigt förlåtande språk. När det kommer att skriva kod handlar det i slutändan mer om vana och smak än restriktiva regler. Ett exempel på hur Objective-C kod ser ut kan ni hitta i Figur 2.6. Koden i figuren visar hur man med Objective-C kan instansiera en array med två strängobjekt och sen skriva ut arrayens innehåll genom iterativa funktioner.

```
// [obj method: parameter];
NSArray *arr= [[NSArray alloc] initWithObjects: @"A", @"B",nil];

//exempel for-loop
for(int i = 0 ; i < [arr count] ; i++){
    NSLog(@"%@", [arr objectAtIndex: i]);
}
//exempel foreach-loop
for(NSString *string in arr){
    NSLog(@"%@", string);
}
```

Figur 2.6: Objective-C kodexempel.

När det gäller grafiska användargränssnitt till Mac OS X och iPhone OS finns det några olika ramverk som är intressanta, det vi använder för utvecklingen av vår applikation är Cocoa. Cocoa kan förklaras som: ett set av objektorienterade ramverk som tillhandahåller den miljö som en applikation under Mac OS X eller iPhone OS befinner sig i [4]. Vanligtvis använder man Objective-C för att komma åt de grafiska objekten man skapar i miljön men man kan även använda språk så som Java [24],Ruby [15] med flera.

2.3.2 Android

I denna sektion kommer vi berätta mer om vad som krävs för att börja utveckla program till Android. Vi kommer gå igenom vilka verktyg som underlättar arbetet och hur programmen kommer ut till användarna.

Eclipse är en utvecklingsmiljö för alla typer av utveckling och språk. Utvecklingsgruppen för Eclipse säger själva att det är: ”ett öppet, utbyggbart IDE för vad som helst men inget speciellt i huvudsak”. Med detta menas att Eclipse tagits fram som ett universalverktyg utan någon speciell målgrupp i åtanke. IDE står för Intergrated Development Environment vilket är ett begrepp som beskriver en mjukvarusvit och ett kraftfullt redskap att arbeta med. Eclipse startade som ett projekt hos företaget IBM som var i akt med att ersätta deras tidigare verktyg för Java-utveckling. Projektet släpptes senare som öppen källkod där det nybildade företaget Eclipse Foundation blev ansvariga för den fortsatta utvecklingen. Men som tidigare definition att det skulle funka för alla typer av språk inte specifikt för Java [34].

Vyer och perspektiv är två begrepp som man stöter på och viktiga definitioner att förstå för att snabbt komma igång med utvecklingsmiljön. Vyer hanteras som fönster vilket kan flyttas runt och som representerar en viss aspekt på uppgift som man vill att vyn ska göra eller informera utvecklaren om. Perspektiv är en uppsättning av vyer som ska hjälpa utvecklaren att ta fram programvara. Detta kan exempelvis vara enen uppsättning av vyer, paket, loggverktyg, texteditor samt syntax-kontroll.

Android Software Development Kit (SDK) [23] innehåller de bibliotek som krävs för att skapa program som kan köras i Android. Likt många andra plattformar i Java [24] använder Android sina egna paket för att skapa komponenter i program med användargränssnitt. Under SDKn har man även alternativet att ladda ner diverse Google APIs, exempelvis Google Maps API [10].

Android Development Tools (ADT) ADT är ett plugin till utvecklingsmiljön Eclipse som har utökat stöd för logg, felsökning, guide för att starta nytt Android projekt, paketering av apk-filer, xml-filer som används för att skapa grafiska användargränsnitt samt simulering av händelser i AVD.

Android Virtual Device (AVD) är en virtuell enhet som simulerar en mobiltelefon. I den simulerade miljön går det att anpassa storleken på skärmen, ge enheten tillgång till internet eller begränsa åtkomsten till enhetens tillgångar. I den simulatorn kan man även välja vilka tillgångar som ska finnas, exempelvis SMS eller internet.

Den virtuella enheten har inte obegränsade resurser att erbjuda när program körs i enheten, allt för att få en så realistisk miljö som möjligt. AVD kan simulera händelser och komplikationer såsom inkommande samtal, sms, GPS-data, e-mail eller ett opålitligt 3G-nät. Eftersom miljön är så verklighetstrogen är det först när man kommer till gränssnitt, design och viss användarinteraktion som det kan bli nödvändigt att flytta ett program till en fysisk enhet.

Dalvik är den virtuella maskinen som kan köra och kompilerad Java-kod efter att det konverterats till formatet Dalvik Executable (.dex). Dx heter verktyget som skapar dex-filer ifrån klass-filer, som innehåller klasser och variabler. De variabler som förkommer på flera ställen inkluderas endast en gång vilket resulterar i att dex-filer blir några procent mindre än jar-filer som är Javas motsvarighet för att paketera flera filer till en fil.

Valet mellan stackbaserade eller registerbaserad maskiner är något som ständigt diskuteras. Androids Dalvik VM tillämpar register-baserade arkitektur vilket gör att den skiljer sig från de vanligaste JVM (Java Virtuell Maskin) som implementerar stackmodellen. Stackmodellen kräver kontinuerliga instruktionsuppdateringar för att ladda och manipulera data på stacken. Medan registermodellen inte kräver lika mycket instruktioner för att implementeras. Däremot blir filerna större med registermodellen eftersom den måste kompilera både käll och destinations -register. Fördelarna blir således med register-

baserad att kostsamma operationer som består av främst maskin instruktioner blir färre än stack-modellen, men att storlek på applikationerna blir större för register-baserad modell [32].

Kapitel 3

Neverlost Calendar

Detta avsnitt beskriver vad vi har gjort, varför vi gjorde det och överskådligt vilka komponenter som krävdes. Avsnittet kommer beröra schemasystemet Neverlost, kalender-tjänsten Google Calendar samt Google Data API. Avsnittet beskriver även de protokoll och filtyper som används i implementationen av vår applikation.

3.1 Bakgrund

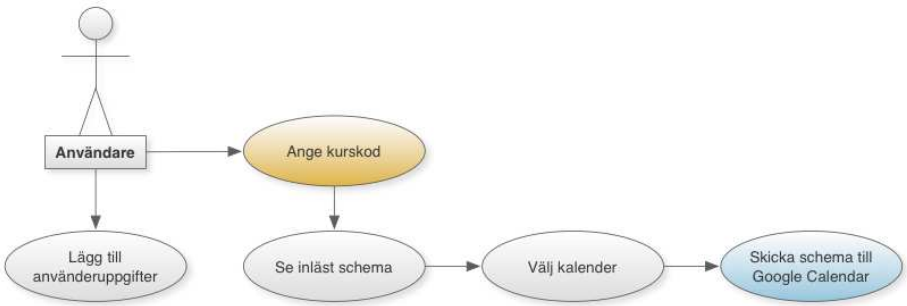
Användningen av datorbaserade kalendrar har blivit allt mer populärt och finns nu integrerat i de flesta mobiltelefoner. Precis som många andra telefoner på marknaden har iPhone och Android dessutom möjligheten att synkronisera kalendern på mobiltelefonen med kalendrar antingen på en avlägsen server eller de man har på sin persondator. Vi använder oss båda av Google Calendar för att hålla ordning och reda på våra liv och synkroniserar den kalendern med våra mobiltelefoner.

Karlstads universitet använder i sin tur ett system som heter Neverlost för att schemalägga kursrelaterade händelser. Schemat för varje kurs finns ofta på kurshemsidorna i form av en länk till Neverlost där studenterna kan få fram schemat i HTML-format. Vi är så pass vana med att använda våra kalendrar och att dagens aktiviteter finns direkt i

mobilen. Därför tyckte vi att det nuvarande systemet var klumpigt och otillgängligt. Som lösning på detta problem utvecklade vi en applikation som ger användaren möjligheten att ladda upp Neverlost-scheman till kalendertjänsten Google Calendar.

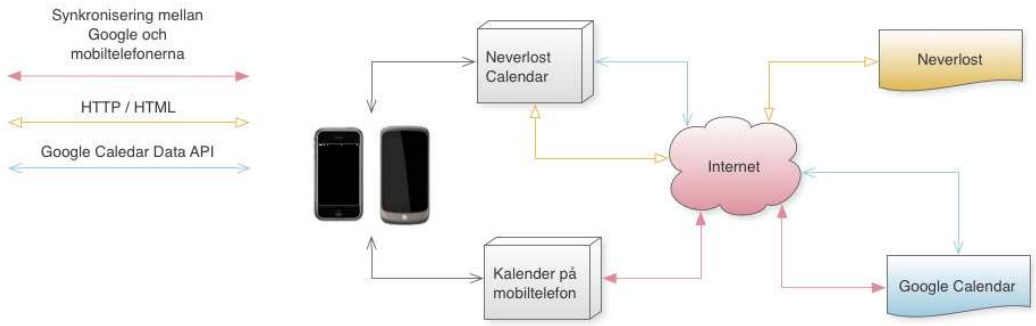
3.2 Motivering

Idén bakom att koppla Neverlost till kalendrar är något som redan har implementerats. I de applikationer som vi fått inspiration från (JNeverlost [22] och Neverlost 2 iCalendar [6]) så anger man först en kurskod, sedan upprättar man en HTTP-session mot Neverlost då man hämtar ner ett HTML-dokument med kursinformation som sedan tolkas och omvandlas till en standardiserad kalenderfil.



Figur 3.1: Användardiagram över Neverlost Calendar.

Med applikationerna vi utvecklat kan man på samma sätt ange en kurskod och hämta den angivna kursens schema genom Neverlost. Men istället för att generera en kalenderfil så kommer man först att ange sina användaruppgifter till Google Calendar för sedan att kunna välja att lägga in schemats händelser i en kalender. När detta är gjort kan man i telefonen under inställningarna för kalender lägga till Google-kalendern på valfritt sätt. Diagrammet i Figur 3.1 illustrerar hur användaren interagerar med vår applikation för att utföra denna uppgift. Figur 3.2 illustrerar däremot alla kopplingar, från vår applikation till synkroniseringen med användarens mobiltelefon.



Figur 3.2: Kopplingen mellan Google Calendar och applikationen Neverlost Calendar.

Anledningen till att vi skickar schemat till Google Calendar istället för att lägga in det direkt i telefonens förinstallerade kalender är att vi vill förespråka konceptet med att lagra data på en avlägsen server. Genom att göra detta istället för att endast lagra datan lokalt blir vi mindre bundna till våra personliga maskiner och det ger oss möjligheten att synkronisera olika enheter med varandra. Till exempel är denna uppsats lagrad på en tjänst som heter Dropbox [8]. Dropbox ger användaren möjlighet till att lagra data på sin enhet men även på en server som kontinuerligt uppdateras samt versionshanteras. Vi har dessutom delat en katalog mellan varandra och får på så vis direkt när filen sparas ta del av varandras ändringar. Med andra ord har vi också hela tiden minst tre backuper på filerna samt att vi kan komma åt dessa via våra mobiltelefoner.

3.3 Komponenter

För att kunna utveckla denna applikation måste vi förstå oss på hur Neverlost fungerar och hur man använder Googles API:er. API står för Application Programming Interface och detta begrepp innebär att man på ett förenklat sätt kopplat ett antal mindre funktioner för att utföra en större och mer komplicerad uppgift. Följande avsnitt kommer förklara vad Neverlost är och hur vi kommer använda Google Calendar API. Avsnittet kommer dock börja med att beskriva de filtyper och protokoll som används i implementationen av vår

applikation.

3.3.1 Filtyper och protokoll

eXtensible Markup Language (XML) är ett set av grammatiska regler till för att beskriva elektroniska dokument och dess innehåll [27]. Även om dess huvudsakliga syfte är att beskriva dokument används det i stor utsträckning för att representera hierarkiska datastrukturer. Ett exempel på en sådan struktur beskrivs med koden i Figur 3.3.

```
<bookcollection>
    <book type="pocket">
        <title>1984</title>
        <author>George Orwell</author>
    </book>
    <book type="bound">
        <title>Douglas Adams</title>
        <author>The Hitchhiker's Guide to the Galaxy</author>
    </book>
</bookcollection>
```

Figur 3.3: XML kodexempel.

Koden Figur 3.3 visar hur man kan göra för att strukturera information om en boksamling. I så kallad välformulerad XML-kod måste alla element vara omslutna av ett rotelement (<bookcollection>) samt att alla element (<book>) måste avslutas med en slut-tagg (</book>). Alla element kan dessutom få attribut (type="pocket") för att ytterligare tydliggöra elementets innehåll (exempel George Orwell).

För att läsa dokumentet och dess innehåll implementerar man en så kallad XML-tolk. Genom att programmeringsmässigt först definiera reglerna för XML kan men genom stränghantering leta efter mönster som håller sig enligt de grammatiska reglerna. Exempelvis läser man in ett element genom att först leta efter karaktären <. Efter < bör det sedan komma ett antal alfanumeriska tecken så som "book" som i sin tur avslutas med karaktären >. Allt eftersom man läser in element och dess innehåll kan man exempelvis skapa ett träd av noder. I trädet representerar varje nod ett element och dess tillhörande

attribut och innehåll.

HyperText Markup Language (HTML) är precis som XML ett set av grammatiska regler till för att beskriva elektroniska dokument och dess innehåll. Till skillnad från XML är HTML till för att grafiskt representera dess egna innehåll genom en webbläsare [26]. För att webbläsaren ska kunna rita upp dokumentets innehåll på ett korrekt sätt måste HTML-koden innehålla särskilda element. Figur 3.4 visar ett exempel på vilka element ett HTML-dokument, som minst, måste innehålla.

```
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

Figur 3.4: HTML kodexempel.

Koden i Figur 3.4 ser vi bland annat elementet (en så kallad tagg) <html>. Detta är rottaggen som definierar dokumentet som ett HTML-dokument. Direkt efter html-taggen deklareras huvudet på sidan med taggen <head>. Denna tagg kan genom ett flertal element beskriva dokumentets innehåll, bland annat med taggen <title> som deklarerar titeln på dokumentet. Inom taggen <body> kommer sedan själva innehållet, vilket i det här fallet finns det endast en paragraf (<p>) som skriver ut texten ”Hello World!”.

Hypertext Transfer Protocol (HTTP) är ett klient-server-protokoll som används för att överföra information via ett nätverk [9]. Protokollet implementeras som en separat server och klient. Serverns uppgift är att svara på förfrågningar från klienten och sedan skicka tillbaka information. Förfrågningarna sker genom att klienten skickar en fil till servern innehållande ett antal parametrar. Bland annat kan dessa parmetrar bestå av GET och

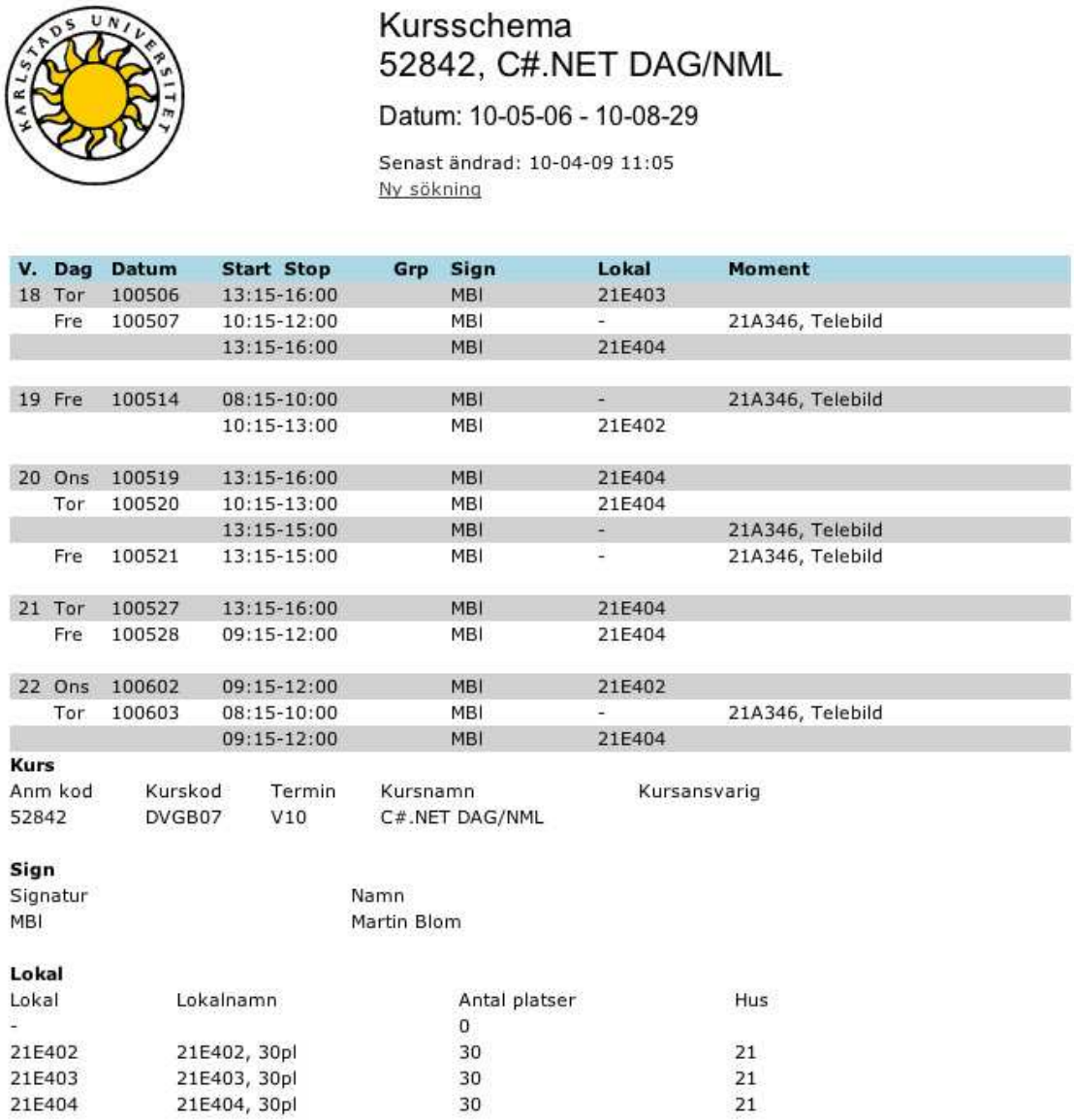
PUT -kommandon. GET-kommandot används till att hämta information medan PUT-kommandot används för att skicka information. Informationen i sig består ofta av filer såsom HTML eller XML -dokument.

3.3.2 Neverlost

Neverlost är det system som Karlstad universitet använder för att exempelvis lägga scheman, boka lokaler, hantera utbildningsprogram och kurser [16]. Systemet bygger på en databas med information som lockas ut med ett ASP-script. ASP är ett språk skapat för att dynamiskt visa innehåll på en webbsida. ASP-koden skriven för Neverlost kompileras på servern och genererar ett HTML-dokument med information tagen från Neverlosts databas.

För att användaren ska komma åt informationen i Neverlosts databas går denne genom en webbportal (http://neverlost.kau.se). Där kan användaren få ut scheman för enstaka kurser eller hela avdelningar. Hur detta kan se ut visar bilden i Figur 3.5. Bilden visar schemat för kursen C#.NET på Karlstad Universitet. I schemat finns bland annat information om föreläsningar så som dag, tid, plats, föreläsare samt moment. Längre ner i bilden finns även information om kursansvarig, vilket hus lokalerna tillhör och hur många platser det finns i respektive lokal.

Applikationen är utvecklat på Högskolan i Borås som nu har lagt ner projektet för vidareutveckling och arbetat vidare med ett annat projekt.



Kursschema
52842, C#.NET DAG/NML

Datum: 10-05-06 - 10-08-29

Senast ändrad: 10-04-09 11:05
Ny sökning

V.	Dag	Datum	Start	Stop	Grp	Sign	Lokal	Moment
18	Tor	100506	13:15	16:00		MBl	21E403	
	Fre	100507	10:15	12:00		MBl	-	21A346, Telebild
			13:15	16:00		MBl	21E404	
19	Fre	100514	08:15	10:00		MBl	-	21A346, Telebild
			10:15	13:00		MBl	21E402	
20	Ons	100519	13:15	16:00		MBl	21E404	
	Tor	100520	10:15	13:00		MBl	21E404	
			13:15	15:00		MBl	-	21A346, Telebild
	Fre	100521	13:15	15:00		MBl	-	21A346, Telebild
21	Tor	100527	13:15	16:00		MBl	21E404	
	Fre	100528	09:15	12:00		MBl	21E404	
22	Ons	100602	09:15	12:00		MBl	21E402	
	Tor	100603	08:15	10:00		MBl	-	21A346, Telebild
			09:15	12:00		MBl	21E404	

Kurs

Anm kod	Kurskod	Termin	Kursnamn	Kursansvarig
52842	DVGB07	V10	C#.NET DAG/NML	

Sign

Signatur	Namn
MBl	Martin Blom

Lokal

Lokal	Lokalnamn	Antal platser	Hus
-		0	
21E402	21E402, 30pl	30	21
21E403	21E403, 30pl	30	21
21E404	21E404, 30pl	30	21

Figur 3.5: Skärmdump av Neverlost.

3.3.3 Google Calendar Data API

Google Calendar är en tjänst till för att samla händelser med syftet att förenkla planeringen av en användares dagliga aktiviteter. För att sprida tjänsten har Google utvecklat ett så kallat API (Application programming interface) [31] som möjliggör annan mjukvara att

använda sig av tjänsten. Genom APIt kan olika operationer genomföras såsom att hämta, ta bort eller uppdatera händelser. Applikationen Neverlost Calendar implementerar detta API för att kommunicera mellan telefonen och Google Calendar [11].

Kommunikationen mellan det implementerade APTt och Google Calender sker genom att skicka XML-dokument med händelseinformation över en HTTP-session. I Figur 3.6 ser vi ett exempel på hur XML-koden kan se ut.

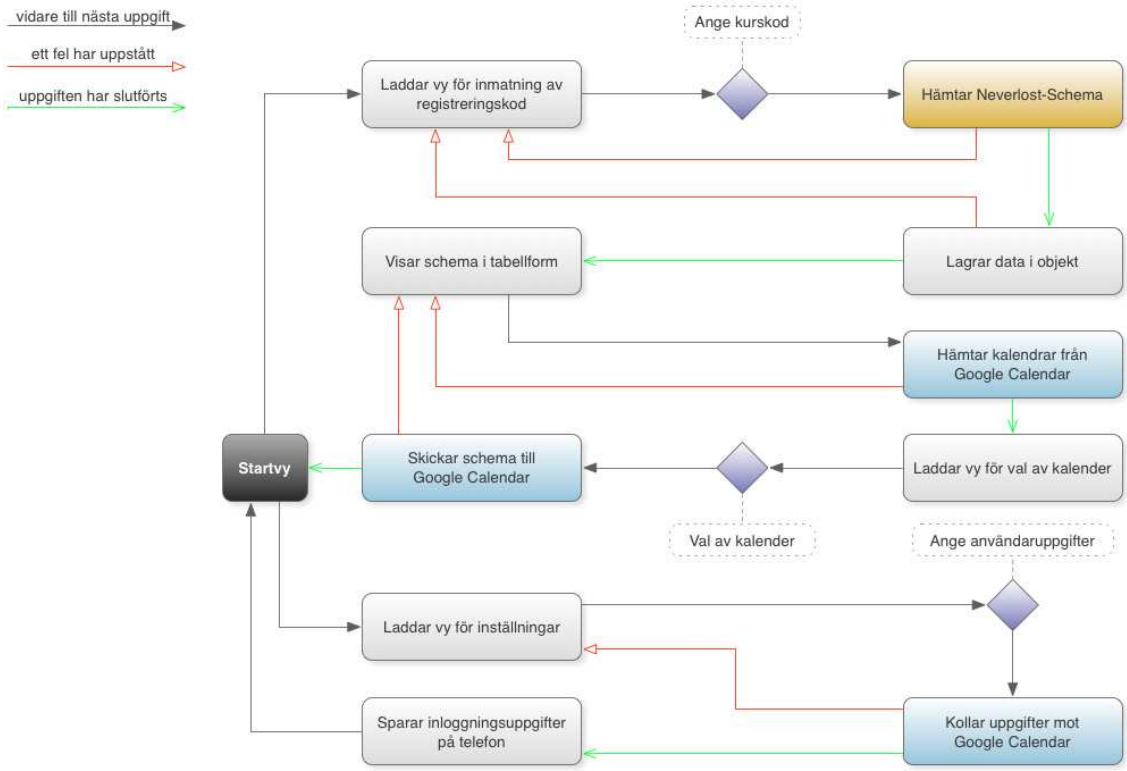
```
<entry xmlns='http://www.w3.org/2005/Atom'
    xmlns:gd='http://schemas.google.com/g/2005'>
  <category scheme='http://schemas.google.com/g/2005#kind'
    term='http://schemas.google.com/g/2005#event'></category>
  <title type='text'>Tennis with Beth</title>
  <content type='text'>Meet for a quick lesson.</content>
  <gd:transparency
    value='http://schemas.google.com/g/2005#event.opaque'>
  </gd:transparency>
  <gd:eventStatus
    value='http://schemas.google.com/g/2005#event.confirmed'>
  </gd:eventStatus>
  <gd:where valueString='Rolling Lawn Courts'></gd:where>
  <gd:when startTime='2006-04-17T15:00:00.000Z'
    endTime='2006-04-17T17:00:00.000Z'></gd:when>
</entry>
```

Figur 3.6: XML-kod för att lägga till enn händelse genom Google Calendar Data API.

I koden anger typ av händelse (category), i det här fallet en vanlig kalenderhändelse (event). Efter det kommer själva händelseinformationen, det vill säga titel (Tennis with beth), notering (Meet for a quick lesson), plats (Rolling lawn court) och tid (2006-04-17T17:00:00.000Z).

Kapitel 4

Design

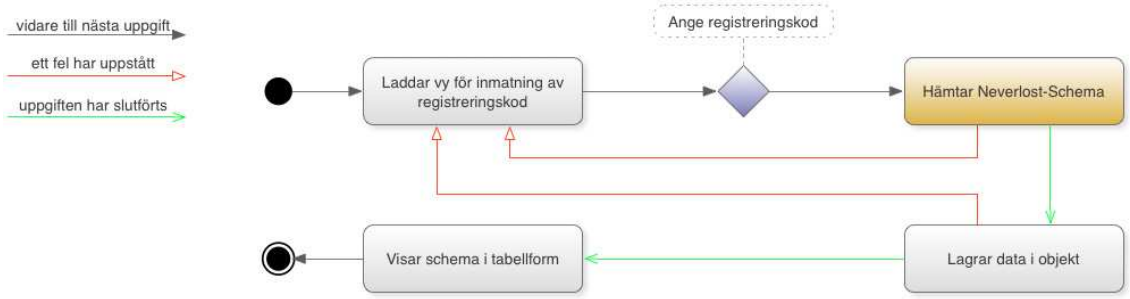


Figur 4.1: Flödesdiagram över alla delar av applikationen.

I föregående kapitel beskrevs vad vi har gjort och varför. I detta kapitel förklaras mer detaljerat hur vi har genomfört vår idé, vilken funktionalitet som finns och hur alla delar hänger ihop. Figur 4.1 är ett diagram över den funktionalitet som implementerats samt flödet mellan de olika delarna av applikationen. I det följande kapitlet kommer Figur 4.1 att delas upp och beskrivas i olika steg. Det första steget var att skapa funktionalitet för att hämta ett angivet schema från Neverlost som sedan tolkas och läggs in i objekt. Under det andra steget implementerades stöd för att spara användaruppgifter till Google Calendar. I det sista och slutgiltiga steget fick applikationen funktionaliteten att välja kalender samt ladda upp schemat till Google Calendar.

4.1 Hämta och tolka schema

En av de huvudsakliga uppgifterna som applikationen har är att hämta ett schema från Neverlost och visa det. Hur detta fungerar och vad som kommer att beröras i detta avsnitt representeras med diagrammet i Figur 4.2. Kort beskrivet så visar diagrammet hur applikationen tar emot en kod som representerar en viss kurs. Sedan kontrolleras koden om den är korrekt eller inte. Om den är korrekt kommer schemat att lagras i objekt som sedan representeras i en tabellvy. Om koden inte är korrekt kommer däremot programmet återgå till vyn för inmatning av kod.



Figur 4.2: Flödesdiagram över att hämta Neverlost schema.

Karlstad Universitet har två olika koder som identifierar en kurs. Den ena är en så kallad kurskod (exempelvis DVGC07) och den andra en registreringskod (exempelvis 52842). Kurskoden identifierar själva kursen, vad den behandlar, när den går och så vidare medan registreringskoden identifierar kursen vid ett visst kurstillfälle. Neverlost behandlar endast registreringskoden, däremot kommer studenten endast i kontakt med denna när hon/han registrerar sig. Detta innebär att när studenten/användaren söker på en kurs så måste en kontroll utföras före det att schemat laddas ner.

http://www.neverlost.kau.se/schema.asp?Start=&kursId=kau.52842&v=1&typ=1&fk=1

Figur 4.3: Exempel på GET-url till Neverlost.

Det enda sättet att hämta ett schema är genom protokollet HTTP [9]. För att hämta rätt schema krävs en URL-sträng [45] som pekar mot Neverlost-servern samt ett antal GET-parametrar [35] som beskriver vilket schema man vill visa. I Figur 4.3 kan man se hur URL-strängen är uppbyggd. I figuren ser vi även kursens registreringskod som anges med GET-kommandot kursid. Denna kod är det enda som förändras i strängen mellan sökningarna med applikationen. I URL-strängen kan vi kan även bestämma hur mycket information vi vill ha, exempelvis från vilket datum schemat ska visa händelser. Detta anges med GET-kommandot Start. I Figur 4.3 har Start inget värde, vilket betyder att alla händelser från och med kursens start kommer att hämtas. När URL-strängen är skapad så upprättas en HTTP-session med en förfrågan mot servern som skickar tillbaka ett HTML-dokument. Eftersom Neverlost endast kan generera ett HTML-dokument så måste dokumentets data tolkas innan vi kan använda schemats informationen i vår applikation. För att kunna tolka informationen så rensar vi HTML-dokumentet från onödig information så att det blir välformulerat nog för att köras genom en XML-tolk. XML-tolken skapar ett träd av noder som vi sedan använder för att extrahera information, så som händelser och kursinformation, för att sedan lägga in informationen i objekt.

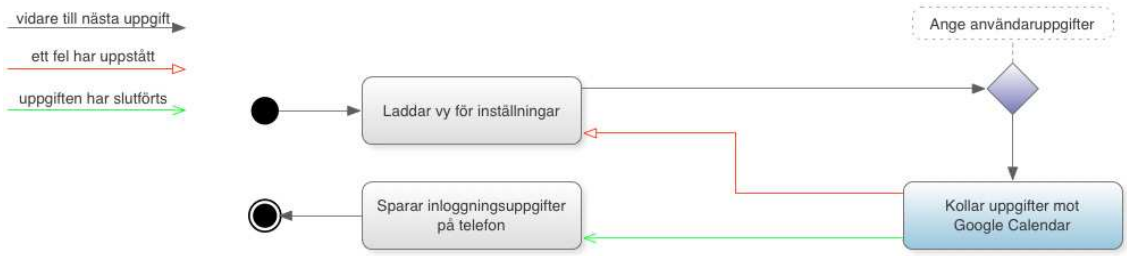
Figur 4.4: Klassdiagram över objektschema.

Figur 4.4 är ett klassdiagram över objekten som tillfälligt lagrar schema-datan i minnet. Klassdiagrammet består av 3 olika klasser, nämligen NeverlostHandler, Schedule och Event. NeverlostHandler är den klassen som hanterar tokningen av HTML-dokumentet samt instansierar ett objekt av klassen Schedule. Schedule i sin tur lagrar informationen i olika Event-objekt.

Objekten som skapat kan sedan användas till att enkelt visa schemat med en tabellvy i vår applikation samt skicka informationen till Google Calendar. Detta beskrivs mer detaljerat i det sista avsnittet.

4.2 Användarinställningar

Med applikationen kan man nu ange en kod som tillhör en kurs, hämta den kursens schema samt visa upp schemat i en tabell. Före schemat kan läggas upp på Google Calendar så måste användaren ange sin uppgifter till Google Calendar och spara dom. Detta avsnitt går igenom denna procedur.



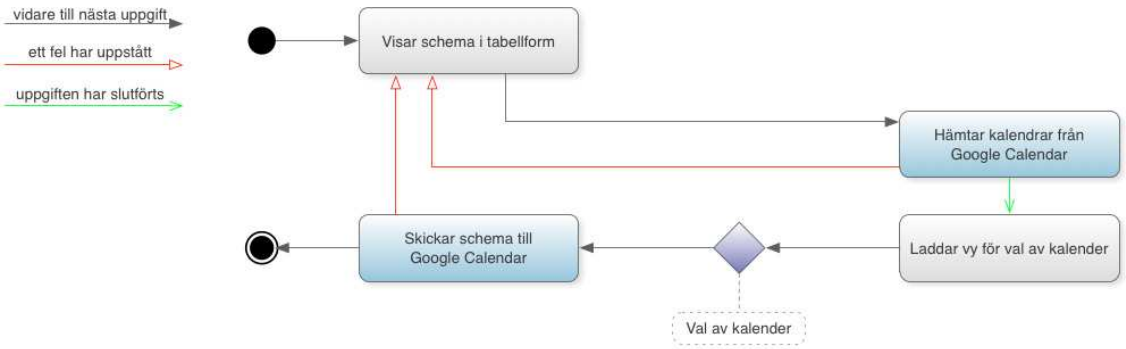
Figur 4.5: Flödesdiagram över användarinställningar.

För att applikationen ska vara användbar för vem som helst måste vi också kunna välja vilket konto på Google som schemat ska laddas upp till och även vilken kalender den ska ligga på. Därför har vi skapat en vy för inställningarna där man ska man ange, kontrollera och sedan spara inloggningsuppgifterna på telefonen för framtida bruk. Inloggningsuppgifterna består av det användarnamn och lösenord man har fått av Google för att logga in på Google Calendar.

I Figur 4.5 laddar användaren vyn för inställningar där man matar in användarnamn och lösenord. När användaren sedan sparar uppgifterna görs en koll mot Google Calendar där applikationen försäkrar sig om att uppgifterna stämmer. Om kontrollen går igenom skickas användaren tillbaka till huvudvyn, om inte så stannar man kvar i inställningsvyn och får ett felmeddelande.

4.3 Skicka till Google Calendar

Nästa steg i utveckling var att kunna lägga upp sitt nedladdade schema i en kalender på Google Calendar. I Figur 4.6 illustreras hur användaren direkt från schemavyn kan gå vidare till att publicera sitt schema.



Figur 4.6: Flödesdiagram över när datan skickas till Google Calendar.

För att försäkra oss om att ingenting går snett och att alla uppgifter stämmer så kommer applikationen att varje gång man publicerar ett schema kolla uppgifterna mot Google Calendar. När väl användaren har valt en kalender och trycker vidare för publicering så kommer schemat att läggas upp på den angivna kalendern. Om något går fel kommer dock användaren att skickas tillbaka till tabell-vyn och meddelas om felet. Meddelande i sig kan bero på uppkopplingssvårigheter eller att datan av någon anledning är felaktig.

Kapitel 5

Användargränssnitt

Det följande kapitlet kommer att behandla hur funktionaliteten i applikationer representerade med hjälp av GUIs, hur användaren interagerar med telefonerna och vilka datakontroller med felmeddelanden som är implementerade. I kapitlet förklaras även vilka riktlinjer som legat till grund för designen av GUIt.

5.1 Riktlinjer

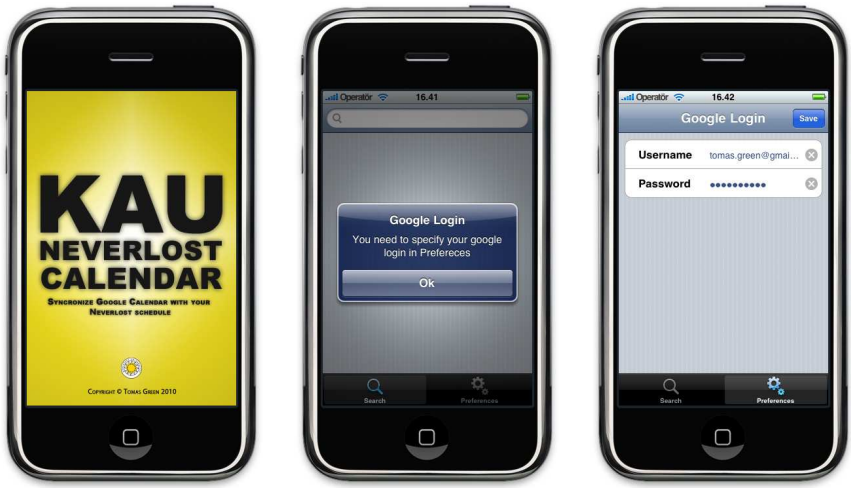
När vi började utveckla bestämde vi oss för att hålla oss till vissa riktlinjer. Under arbetet med designen av applikationen ville vi använda iPhones [3] och Androids [21] standardiserade utseende. Anledningen var att att applikationen skulle smälta in i operativsystemets miljö och att användaren skulle känna sig hemma i navigeringen och användandet av applikationen.

5.2 iPhone

Följande avsnitt kommer handla om hur GUIt ser ut och fungerar i iPhone-implementationen. Vi kommer att beröra nästan alla delar av hur applikationen ser ut med undantag för vissa

felmeddelanden. De delar vi kommer att beröra är: uppstart och inställningar, sökning av kurser och publicering av schemat till Google Calendar.

5.2.1 Uppstart och inställningar



(a) Splashscreen (b) Uppdatera inställningarna (c) Inställningar

Figur 5.1: Uppstart och inställningar.

Det här avsnittet kommer handla om hur applikationen fungerar vid uppstart och vad som händer när man anger sina kontouppgifter för Google Calendar. Först och främst, när man utvecklar en applikation kan man välja att sätta en så kallad splashscreen som visas under den tiden som applikationen laddas in i minnet, Figur 5.1(a) visar hur iPhone-applikationens splashscreen.

När applikationen väl laddats in i minnet så sker en kontroll hurvida det finns några konto-inställningar för Google Calendar. Figur 5.1(b) visar första gången man startat applikationen, det vill säga innan användaren angett några kontouppgifter. När användaren tryckt på inställningar (Preferences-tabben), fyllt i fälten och tryckt på save-knappen, precis som i Figur 5.1(c), så kommer applikationen åter igen att kolla om inställningarna

stämmer genom att upprätta en koppling mot Google Calendar. Under tiden som kopplingen upprättas så tonas gränssnittet ut för att sedan återgå till samma läge som i Figur 5.1(c). Om det inte fungerar kommer dock användaren att få upp en ruta med ett felmeddelande.

5.2.2 Sökning av kursscheman

(a) Kodformat (b) Kodfel (c) Sökresultat

Figur 5.2: Sökning av kursscheman.

Följande avsnitt kommer att beskriva hur det fungerar när man söker på kurser. Eftersom det finns två olika koder för varje kurs varav endast en fungerar med Neverlost så måste applikationen avgöra om användaren har angett en korrekt skriven kod.

I Figur 5.2(a) finns ett exempel på när användaren vill hämta schemat för kursen C#.NET vars kurskod är DVGB07. Denna kod är dock inte kompatibel med Neverlost och därför får användaren ett meddelande om att slå in den korrekta koden, det vill säga registreringskoden. Om användaren däremot har skrivit in en felaktig registreringskod, det vill säga en kod som inte har någon tillhörande kurs eller vars schema ännu inte existerar så kommer användaren än en gång varnas. Ett exempel på detta ser vi i 5.2(b).

I Figur 5.2(c) där användaren har skrivit in en korrekt kod så visas resultatet direkt i

tabellform . Uppe i högra hörnet kommer nu en synk-knapp att aktiveras med kravet att en uppkoppling mot Google Calendar kan upprättas.

5.2.3 Val av kalender och ändring av titel

(a) Valmöjligheter (b) Ändra kalender (c) Ändra titlar

Figur 5.3: Val av kalender och ändring av titel.

Användaren har nu angett korrekta kontouppgifter, sökt efter en kurs och fått ner det tillhörande kursschemat och kan nu fortsätta med att lägga upp schemat på Google Calendar.

Figur 5.3(a) visar hur skärmen ser ut direkt efter att man tryckt på sync. Här kan man nu välja kalender och ändra schema-händelsernas titlar. De kalendrar man kan välja mellan är de som finns tillgängliga under sitt Google Calendar-konto. Figur 5.3(b) visar hur det ser ut när man trycker på Select calendar". Rullhjulet som visas är standard för iPhone-applikationer och som är jämförbar med en vanlig drop-down lista som finns i de flesta operativsystem eller webbformulär. När detta rullhjul laddas så kommer allting bakom skärmen att skuggas ut och göras otillgänglig för användaren tills denne har valt kalender.

Användaren har som sagt även möjlighet att ändra titel på samtliga händelser och ett exempel på detta ser vi i Figur 5.3(c).

5.2.4 Uppladdning till Google Calendar

(a) Uppladdning till Google Calendar

(b) Uppladdning avslutad

Figur 5.4: Uppladdning till Google Calendar.

För att påbörja eller avsluta uppladdningsprocessen trycker användaren på knappen Publish respektive Cancel. Om användaren skulle ändra sig om att lägga in schemat på sin kalender och trycker Cancel så kommer denne slussas tillbaka till sökresultatet. Det som händer om användaren trycker på Publish är att alla poster i tabellen läggs in i den valda kalendern på Google Calendar. Under tiden detta sker kommer hela vyn att låsas för användarinteraktion. Figur 5.4(a) är ett exempel på hur detta ser ut. Om nu alla händelser har lagts in utan att något gått fel så kommer man få upp ett verifikationsmeddelande precis som i Figur 5.4(b).

5.3 Android

Det här avsnittet handlar om hur användargränssnittet för Androidapplikationen ser ut, vilka vyer som som finns och användaren interagerar med de dessa. Avsnittet kommer beröra allt från före uppstart av applikationen till resultatet av en uppladdning till Google Calendar.

5.3.1 Uppstart och inställningar

(a) Launcher (b) Startup (c) Startup

Figur 5.5: Uppstart och inställningar.

Figur 5.5(a) är ett exempel på hur Android hemskärm kan se ut före man startat vår applikation. I figuren kan vi även se den ikon som representerar applikationen. Denna ikon är precis som resten av applikationen skapad utifrån särskilda riktlinjer [25].

Det första användaren kommer i kontakt med efter att applikationen har startat är den vy som visas i Figur 5.5(b). I vyn uppmanas användaren att välja ett Google-konto. De konton som visas i figuren är hämtade från de synkroniseringsprofiler som användaren har lagt till genom inställningarna i Android OS. När väl användaren valt en profil kommer

denna, precis som i Figur 5.5(c) uppmanas att söka efter en kurs. Mer om hur detta fungerar i nästa avsnitt.

5.3.2 Sökning av kursscheman

(a) Skriv anmkod (b) Hämtar schema (c) Resultat av lyckad sökning

Figur 5.6: Sökning av kursscheman.

Användaren söker på kurser genom att matar in en sträng av siffror i sökfältet. Om strängen kan generera ett resultat från Neverlost visas detta i tabellform direkt under sökfältet. Figur 5.6(a) visar vad som händer när sökrutan är markerade och användaren skriver in koden. Efter detta trycker användaren på done och medan applikationen genomför sökningen visas den ruta vi kan se i Figur 5.6(b). Denna ruta är synlig tills dess att hela schemat laddats ner, tolkats och matats in i den tabellvy som syns i Figur 5.6(c).

5.3.3 Val av kalender och ändring av titel

(a) Ändra titel (b) Meny (c) Lista över kalendrar

Figur 5.7: Val av kalender och ändring av titel.

Användaren har nu hämtat schemat och titel för schemat kanske inte passar eller är direkt missvisande. Då går det att ändra titeln till något mer passande som ses i Figur 5.7(a). För att användaren ska kunna lägga upp det nerladdade schemat till Google Calendar måste denna välja kalender. För att göra detta trycker användaren på telefonens menyknapp. I menyn kan nu användaren precis som i Figur 5.7(b) välja fliken Calendars. När knappen är tryckt hämtas en lista över de kalendrar som finns kopplade till den synkroniseringsprofil användaren valde vid uppstart av programmet (se Figur 5.6(b)). I Figur 5.7(c) kan vi se hur denna lista representeras. I listan kan nu användaren med ett fingerdrag välja vilken kalender som schemat ska laddas upp till.

5.3.4 Uppladdning till Google Calendar

(a) Menyknapp för att ladda upp.

(b) Resultat av en lyckad uppladdning.

Figur 5.8: Uppladdning till Google Calendar.

Följande avsnitt kommer beskriva de sista och slutgiltiga vyerna för att kunna ladda upp det nedladdade schemat till en kalender på Google Calendar. Figur 5.8(a) visar hur användaren åter igen tryckt upp menyn valt Upload. Efter att detta är gjort kommer uppladdningen ske i bakgrunden men för att användaren ska vara delaktig i processen så kommer den gråa cirkeln vid varje händelse att ändra färg från grå till grön. Grå illustrerar givetvis att uppladdningen inte lyckades och grön vise versa. Figur 5.8(b) visar hur en helt lyckad uppladdning ser ut.

Kapitel 6

Implementation

Det här kapitlet kommer att beskriva hur applikationen är implementerad för plattformarna iPhone och Andorid. I avsnittet 6.1 Översikt kommer vi att beskriva vad designmönstret Model View Controller är och hur det är tillämpat för våra ändamål. I samma avsnitt kommer vi även beskriva vissa problem med tolkningen den HMTL-kod som Neverlost genererar.

6.1 Översikt

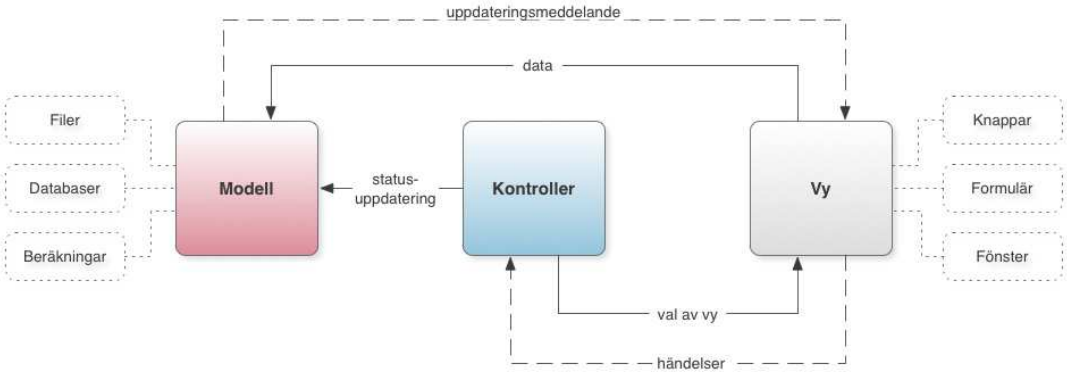
Det här avsnittet kommer att beskriva designmönstret Model View Controller samt förklara tolkningen den HMTL-kod som Neverlost genererar.

6.1.1 Designmönstret Model View Controller

Model View Controller (MVC) [41] är ett så kallat designmöster. Designmönster kan beskrivas som: en problemidentifieringsteknik inom arkitektur och programutvecklingsmetodik som innebär att man katalogiserar olika typiska problem och deras typiska lösningar [33].

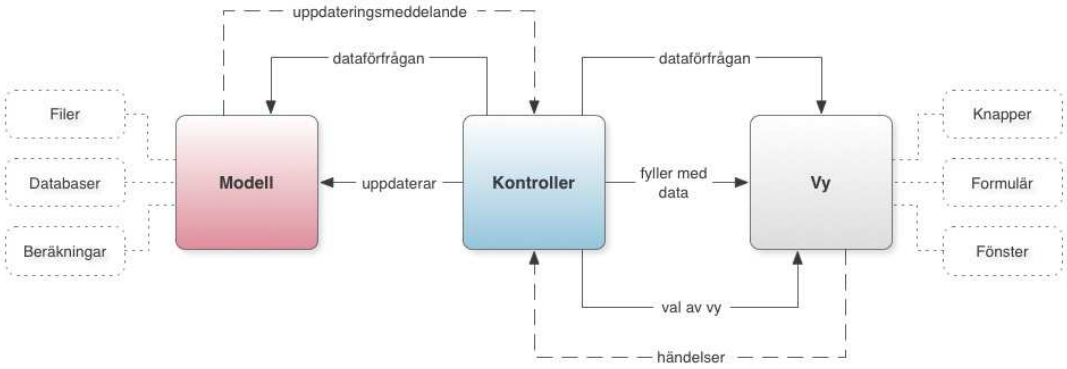
Designmönstret MVC innebär att man särar på vy(view), data/modell (model) samt att man inför en kontroller(controller), som agerar på begäran av vyn och som i sin tur

gör förfrågningar till modellen.



Figur 6.1: Diagram över designmönstret Model View Controller.

Figur 6.1 illustrerar hur MVC är menat att fungera. Tanken är att man skapar det GUI man vill ha och kopplar det sedan till en kontroller samt en datakälla. Kontrollern har i sin tur till uppgift att svara på de händelser som skickas från GUIt och bestämma vad som ska hända härnäst, exempelvis att datan i modellen ska manipuleras eller bytas ut. Denna koppling är lite svår att följa vissa gånger, speciellt när det kommer till direktkopplingen mellan vy och data. I våra implementationer sker all koppling till datan via kontroller.

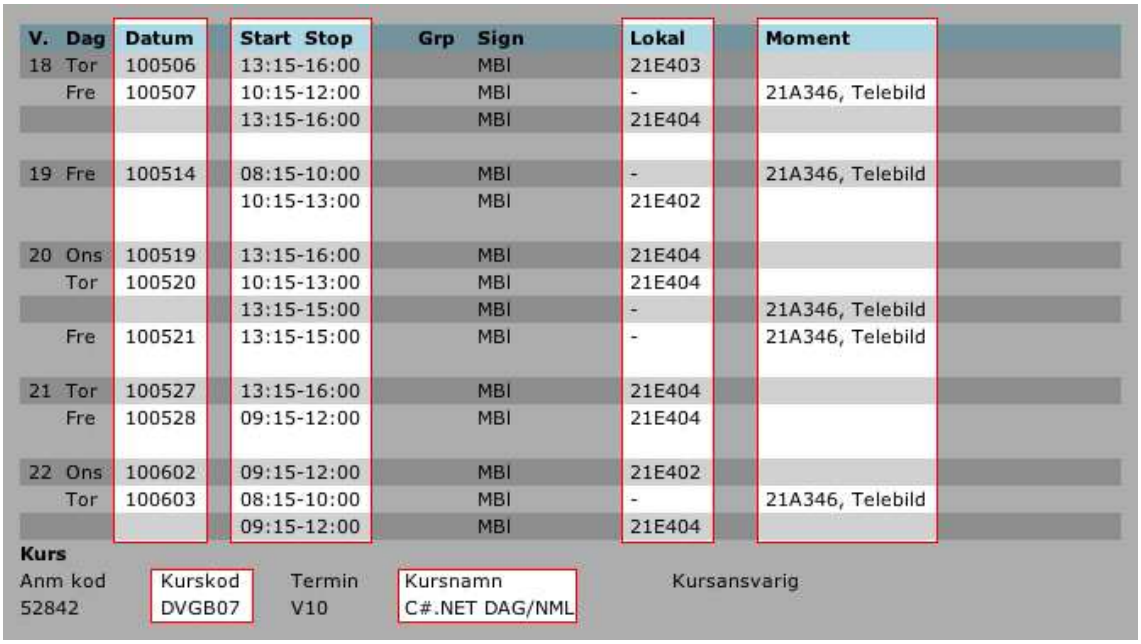


Figur 6.2: Diagram över implementationen av MVC.

Figur 6.2 är ett diagram över den faktiska implementationen av MVC och hur applikationens delar kommunicerar med varandra. Som diagrammet visar har kontrollern en

väldigt central roll. Man skulle kunna säga att kontrollerna är själva applikationen, som i sig håller samman och koordinerar vyerna med modellerna. De följande implementationsavsnitten kommer att beskriva mer detaljerat hur detta är implementerat i de olika plattformslösningarna.

6.1.2 Tolkning av Neverlosts genererade HTML-kod



V.	Dag	Datum	Start	Stop	Grp	Sign	Lokal	Moment
18	Tor	100506	13:15	16:00		MBI	21E403	
	Fre	100507	10:15	12:00		MBI	-	21A346, Telebild
			13:15	16:00		MBI	21E404	
19	Fre	100514	08:15	10:00		MBI	-	21A346, Telebild
			10:15	13:00		MBI	21E402	
20	Ons	100519	13:15	16:00		MBI	21E404	
	Tor	100520	10:15	13:00		MBI	21E404	
			13:15	15:00		MBI	-	21A346, Telebild
	Fre	100521	13:15	15:00		MBI	-	21A346, Telebild
21	Tor	100527	13:15	16:00		MBI	21E404	
	Fre	100528	09:15	12:00		MBI	21E404	
22	Ons	100602	09:15	12:00		MBI	21E402	
	Tor	100603	08:15	10:00		MBI	-	21A346, Telebild
			09:15	12:00		MBI	21E404	

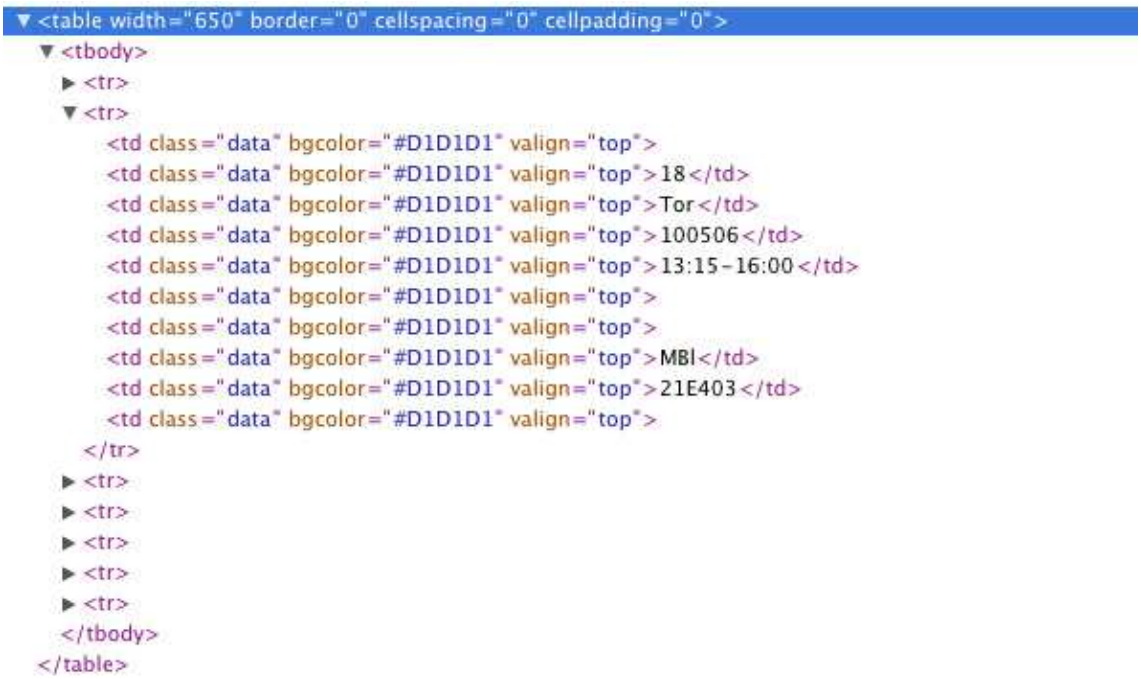
Kurs

Anm kod	Kurskod	Termin	Kursnamn	Kursansvarig
52842	DVGB07	V10	C#.NET DAG/NML	

Figur 6.3: Bild av den data som ska extraheras från Neverlost.

En väldigt viktig del av applikationerna är tolkningen av den HTML-kod som Neverlost har genererar. Figur 6.3 visar vilka fält som är intressanta för att kunna lägga upp schemat på Google Calendar. De första som är markerade, det vill säga datum, start och stop - tid, lokal och moment är de delar som ligger till grund för händelseobjekten som kommer att skapas. I bilden kan vi se att händelserna är markerade med växlande färger på olika rader. Men det finns ett problem som inte syns i bilden. Problemet är att information för samma händelse kan ligga på flera rader i HTML-koden och dessutom med olika antal

kolumner. Hur vi löser detta problem kommer beskrivas senare i kapitlet. Längre ner ser vi ytterligare två markerade fält, ett för kurskod samt ett för titel. Dessa kommer att extraheras och sammanfogas för att användas som titel till varje händelse, exempelvis i formatet ”C#.NET (DVGB07)”.

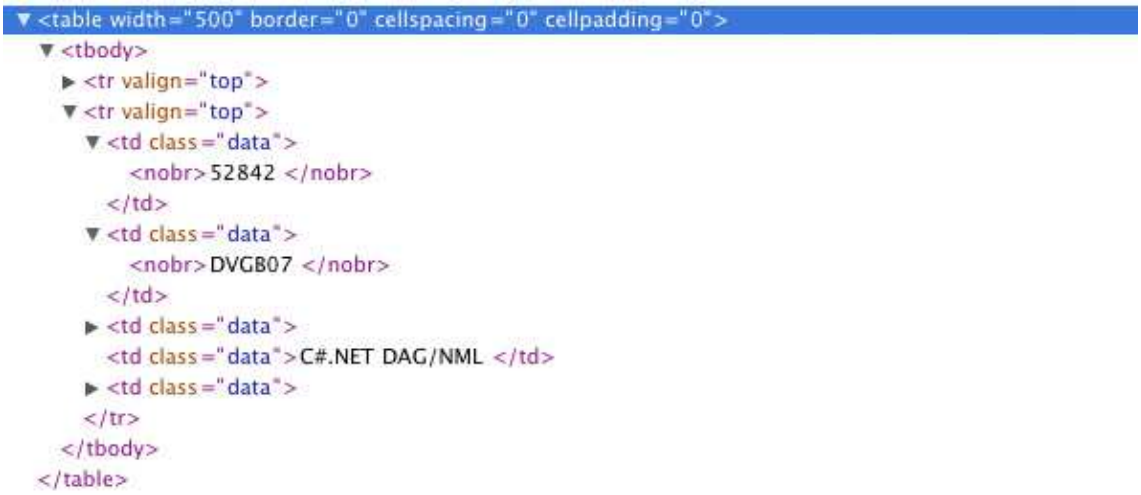


Figur 6.4: Bild av HTML-koden som ska tolkas för händelsedata.

Figur 6.4 visar hur HTML-koden för de första fyra fälten ser ut. Koden genererar en tabell (<table>) med rader (<tr>) och kolumner (<td>). Något att lägga märke till är att ingen av raderna eller kolumnerna går att identifiera på något annat sätt än vilken typ av information det är (class=″ data″). Detta betyder att det inte finns något som identifierar kolumnerna med en viss typ av information. Däremot har varje rad tio kolumner, med andra ord kan vi räkna antalet kolumner för att identifiera vad datan ska tolkas som.

Dock finns det ett antal undantag i tabellstrukturen. Exempelvis kan en händelse i tabellen tilldelas fler än en rad. Anledningen till att detta sker är att någon i efterhand har lagt till extra information i moment-fältet. Problemet som då uppstår är att Neverlost

endast genererar två kolumner varav en sträcker sig över de första nio. Med andra ord kan vi inte längre endast förlita oss på antalet kolumner i en rad när vi plockar ut informationen.



Figur 6.5: Bild av HTML-koden som ska tolkas för titedata.

Nästa bild, nämligen Figur 6.4, visar hur HTML-koden för de sista två fälten i Figur 6.3 ser ut. Det som är intressant att plocka ut ur denna kod är kurskoden (andra raden, andra kolumnen) och kursnamnet (andra raden, fjärde kolumnen).

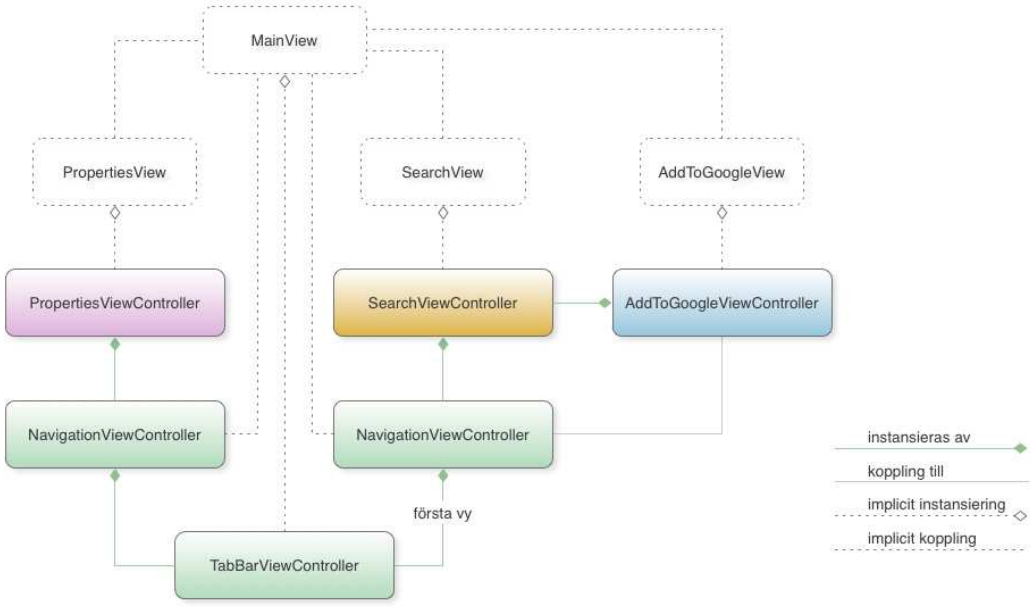
Båda applikationerna använder sig av XML-tolkar för att extrahera informationen. Hur detta fungerar och hur undantagen hanteras är beskrivet i avsnitten för implementationen av iPhone 6.2.2 och Android 6.3.2.

6.2 iPhone

Det här avsnittet beskriver hur applikationen är implementerad för iPhone. Avsnittet börjar med att ge en översikt av applikationens struktur för att sedan delas upp efter de olika designsteg som legat till grund för utvecklingen.

6.2.1 Översikt

För att översiktigt förklara hur vi implementerat applikationen till iPhone har vi ritat upp två diagram. Det första diagrammet i Figur 6.6 visar alla kontrollerklasser och deras kopplingar med respektive vyer. Klasserna längst ner i diagrammet är så kallade navigationsklasser. Dessa ligger i botten av hela applikationen och ansvarar för när och hur en ny vy ska laddas. Diagrammet visar även att finns det två typer av navigationsklasser, en TabBar-klass och två Navigation-klasser. TabBarViewController har en koppling till de knappar som visas längst ner i fönstret i Figur 5.2(c). Däremot är alla titelfält (exempelvis fältet med knappen Upload, längst upp i Figur 5.2(c)) kopplade till en NavigationViewController.

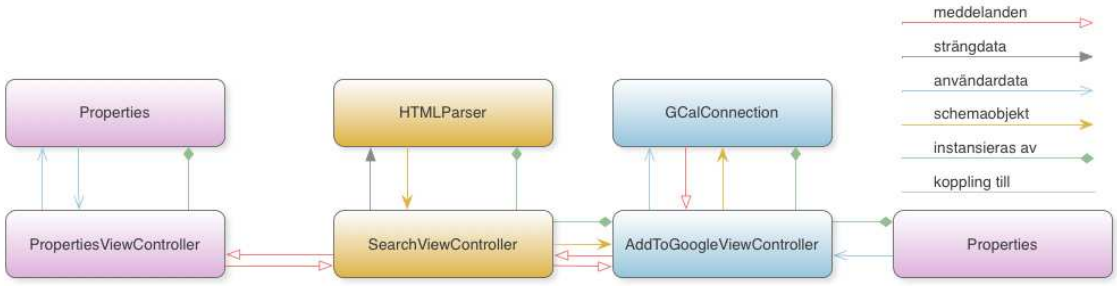


Figur 6.6: iPhone: Diagram över kopplingen mellan vy och kontroller.

Ett exempel på hur det fungerar när en ny vy ska visas är när användaren sökt på en kurs, fått upp resultatet och får möjligheten att trycka på knappen Upload. När användaren tryckt på Upload så kommer SearchViewController att anropa sin NavigationViewController-instans med metoden PushViewController. När detta sker så läggs AddToGoogleViewCon-

troller på en stack och vars tillhörande vy visas på skärmen. Om man sedan trycker bakåt så kommer samma navigations-instans att anropas med metoden PopViewController som tar bort den nuvarande kontrollern från stacken och visar underliggande vy.

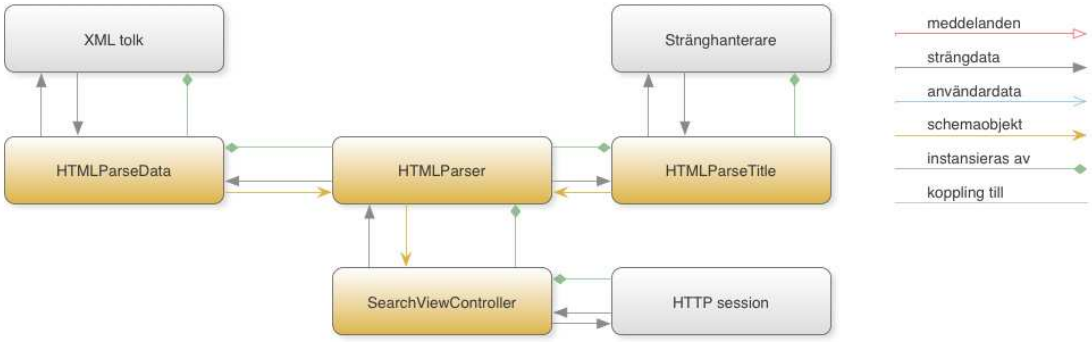
Diagrammet i Figur 6.7 visar kopplingen mellan modell och kontroller -klasser. Figuren visar även hur data och meddelanden flödar mellan de olika klasserna samt hur klasserna instansieras. Däremot är inte alla involverade klasser illustrerade i diagrammet eftersom Figur 6.7 i sin helhet är relativt rörig. Därför kommer de olika delarna att separat beskrivas i de följande avsnitten.



Figur 6.7: iPhone: Diagram över kopplingen modell och kontroller.

6.2.2 Hämta och tolka schema

Efter det att användaren har skrivit in koden för kursen och tryckt på sök så kommer det i SearchViewController att upprättas en HTTP-session mot Neverlost. Session hämtar ner det HTML-dokumentet som Neverlost genererar och lagrar det i en sträng. När väl dokumentet är hämtat så kommer en ny instans av klassen HTMLParser att skapas. Klassen HTMLParser delegerar i sin tur tolkningsfunktionaliteten till två andra klasser, nämligen HTMLParseTitle och HTMLParseData. Detta händelseförlopp illustreras med diagrammet i Figur 6.8. Diagrammet visar hur klasserna instansieras och hur de instanserna skickar data mellan varandra.



Figur 6.8: iPhone: Diagram över hur schemat hämtas och tolkas.

Det första HTMLParser gör är att plocka bort så mycket onödig information som möjligt. Anledningen till det är att HTML-koden vid ett senare tillfälle kommer köras genom en XML tolk, därför måste HTML-koden vara välformulerad. Efter att koden är rensad så hackas dokumentet upp i två delar varav en del innehåller schema-händelser och den andra kurs-information.

För plocka ut titeln ur dokumentet använder sig HTMLParseTitle av strängsökning för att avgöra var titeln för kurser ligger i koden. Titeln sparas sedan i en sträng och kombineras med händelseobjekten som senare kommer att genereras av HTMLParseData.

För att extrahera händelseinformationen kommer det nu skapas en ny instans av HTMLParseData som tilldelas datablocket för händelser. När klassen HTMLParseData instansieras så skapas även en instans av en XML-tolk. XML-tolken läser nu in datablocket för händelser i HTML-dokumentet (se Figur 6.4) och skapa ett träd av noder. Eftersom datablocket är strukturerat i tabellform med rader och kolumner så kommer trädet bestå av en rot-nod, ett X antal radnoder och för varje radnod ett Y antal kolumnnoder.

Nu är dokumentet inläst men för att kunna använda trädet i resten av implementationen måste nodernas information läggas i händelse-objekt (se Figur 4.4). Varje rad i tabellen är en händelse och varje kolumn innehåller den data vi vill komma åt. För varje rad som läses in skapas ett nytt objekt med diverse egenskaper. Vilken egenskap som ska få vilken data hanterar vi genom att räkna antalet kolumner. Beroende på vilket nummer räknaren

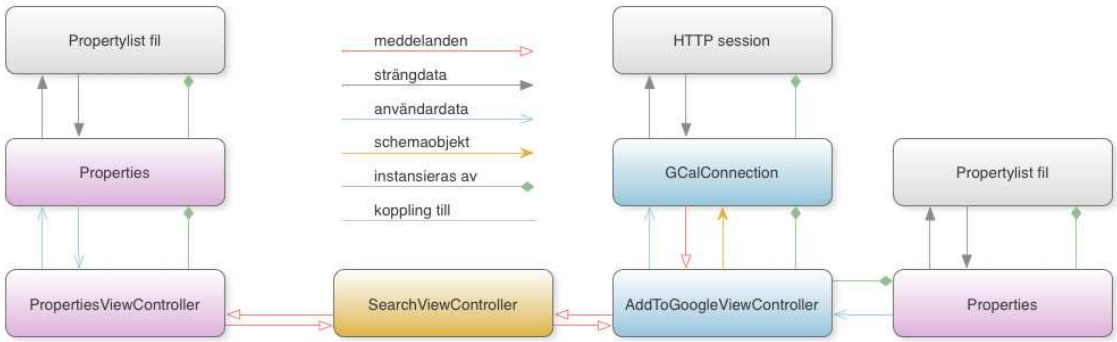
har så läggs kolumnens data in i respektive egenskap.

När det gäller tolkningen av händelse-datan finns det två undantag. Först och främst så kan en händelse tilldelas fler än en rad varav den andra raden innehåller övrig händelseinformation. Problemet är att i den raden finns endast två kolumner. Algoritmen måste ta hänsyn till detta eftersom det inte finns något annat sätt att identifiera vilken tabellkolumn man befinner sig på annat än att räkna antalet kolumner. Om inte så skulle datan i den andra kolumnen att tolkas som ett datum, när det i själva verket är övrig händelseinformation. Som tur är har den första kolumnen på den raden ett attribut som de andra kolumnerna har inte. Om tabellkolumnen man befinner sig på har detta attribut så kombineras kolumnens information med den föregående händelsens info-egenskap.

Ett annat undantag är när flera händelser förekommer under samma dag. Problemet ligger då i att Neverlost inte skriver ut datumet för den händelsen på samma rad. Algoritmen löser detta genom att helt enkelt kopiera den föregående händelsens datum till det nuvarande objektets datum-egenskap.

6.2.3 Användarinställningar

För att spara information använder sig iPhone av något som kallas för propertylist. En propertylist är ett XML-dokument där varje element kan ha ett eller flera attribut och där varje attribut har ett tillhörande värde. Användaruppgifterna till Google Calendar sparas i propertylist med två element, ett för användarnamn och ett för lösenord.



Figur 6.9: iPhone: Diagram över användarinställningarna.

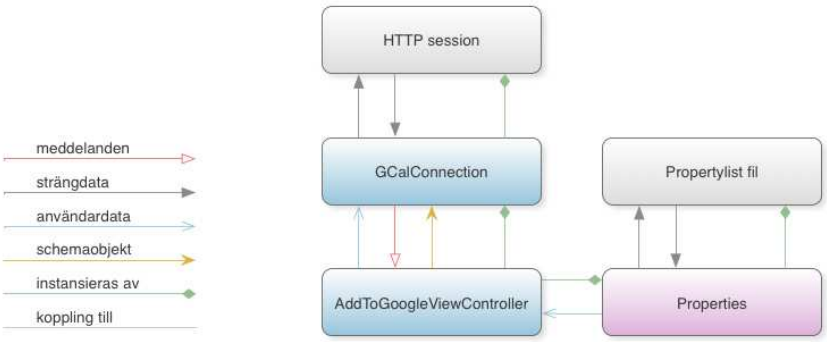
Diagrammet i Figur 6.9 beskriver vad som händer när användaren har skrivit in sitt användarnamn och lösenord samt tryckt på Sparaknappen. När detta sker kommer uppgifterna skickas till en instans av Property som i sin tur skriver uppgifterna till en propertylistfil. Varje gång man skriver in och sparar sina uppgifter så kommer en uppkoppling till Google Calendar att upprättas. Om uppkopplingen lyckas eller inte avgör om uppgifterna stämmer eller ej.

Det finns två anledningar till att uppgifterna sparas före dom kontrolleras. Första anledningen är att uppkopplingen kan vara nere oavsett om uppgifterna är korrekta eller inte. Den andra anledning är att AddToGoogleViewController har en egen instans av Property som används för att hämta uppgifterna och kontrollera om de stämmer. För att det nu ska finnas några uppgifter att använda när man initierar uppkopplingen till Google så måste uppgifterna skrivas till filen före AddToGoogleViewControllers instansierar Property.

När användaren har tryckt på spara så kommer uppgifterna att kontrolleras genom att ett meddelande skickas till AddToGoogleViewController via SearchViewController. AddToGoogleViewController skapar en ny instans av GCalConnection och Property som samverkar för att kontrollera uppgifterna.

6.2.4 Skicka till Google Calendar

Efter att användaren har fått ner sitt önskade schema kan denne gå vidare till uppladdningen av schemat till Google Calendar. För att göra detta trycker användaren på knappen Upload. Då kommer en ny instans av AddToGoogleViewController att skapas, samt ett den nya instansen tilldelas en kopia av schema-objektet. Före vyn visas så kommer SearchViewController att kontrollera om det går att skapa en uppkopplingen till Google Calendar. Om uppkopplingen kan skapas kommer AddToGoogleViewController att läggas på stacken.



Figur 6.10: iPhone: Diagram över publiseringen till Google Calendar.

Figur 6.10 är ett diagram över de kopplingar till de olika klasserna som behövs för att ladda upp schemat på Google Calendar. När användaren väljer att publicera sitt schema så kommer GCalConnection att tilldelas schemaobjektet och påbörja uppladdningsprocessen. För varje Event-objekt i schemat skapas ett GDataEvent. GDataEvent tilldelas därefter datan från ett händelseobjekt och laddas sedan upp till Google. Det som sker när ett objekt skickas till Google Calender är att datan i GDataEvent-objektet läggs in i till en XML-fil. Sedan kommer en HTTP-session öppnas och med hjälp av PUT-kommandot i HTTP skicka den nya XML-filen i en HTTP-förfrågan.

6.3 Android

Det här avsnittet beskriver hur applikationen är implementerad för Android. Avsnittet är uppdelat efter de olika designsteg som legat till grund för utvecklingen.

6.3.1 Översikt

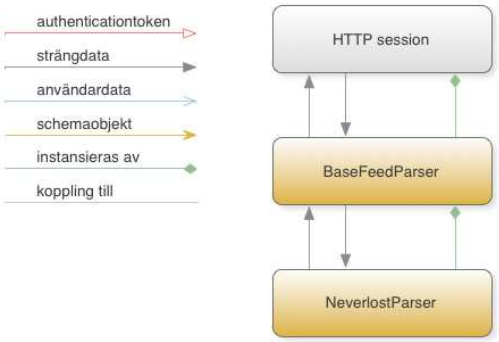
När man implementerar applikationer till Android så täcker gränssnittet hela skärmen vilket tillfälligt begränsar interaktionen mellan användaren och telefonen till den nuvarande startade applikationen. Detta kallas i Android-utveckling för Aktivitet. Aktiviteten har en väldigt central roll i applikationen och som bland annat initierar gränssnittet och agerar som kontroller. Detta innebär att applikationens aktivitet sköter hanteringen av händelser genererat av användarens interaktion som sedan skickar vidare anrop om vilka uppgifter som ska utföras.

I ADT, som beskrevs i Kapitel 2.3.2, ingår ett verktyg som är till för att rita upp gränsnittet direkt i Eclipse. Från början finns de vanligaste komponenterna för att kunna bygga en välfungerande applikation. Passar inte komponenterna för applikationens ändamål går det att med hjälp av verktygen att skapa modifierade komponenter. Komponenter i gränssnittet byggs upp utifrån en XML-fil med attribut tillhörande respektive objekt i gränssnittet.

6.3.2 Hämta och tolka schema

Neverlost genererar HTML-sidor baserade på en URL innehållande ett antal parametrar: kurskod, startdatum samt typ av schema. För att kunna avgöra vad i HTML-koden som är händelsedata så måste applikationen tolka och plocka ut den data som är relevant. Som det går att se i Figur 6.11 har applikationen tillgång till en NeverlostParser. NeverlostParser är i sin tur en utökning av en BaseFeedParser som har till uppgift att förse NeverlostParser med rådatan i HTML-dokumentet.

NeverlostParser har funktionallitet som gör att den kan plocka ut information från schemat enligt en förprogrammerad struktur som baserar sig på hur HTML-koden är uppbyggd. Möter inte uppbyggnaden av HTML-koden upp till de krav som tolken ställer på hur innehållet ska se ut så innebär det att schemat antingen är inaktuellt eller att den inmatade koden inte stämmer.



Figur 6.11: Parsar HTMLsidor som hämtas ner från Neverlost.

I Figur 6.3 kan man se hur schemat är uppbyggda av tabeller i HTML-format. Informationen markerat med rött innehåller kursnamn och de föreläsningar som sker under kursen. Figur 6.4 visar den data som hämtas vid körning. För varje ny rad i tabellen instansieras ett nytt objekt innehållande informationen om händelsen.

Det finns även en del specialfall. Ett av dessa inträffar när ytterligare information till den senaste händelsen hamnar i den kommande raden i tabellen. Då kommer informationen i den kommande raden att läggas till på det föregående objektets informations-egenskap. Som resultat får vi en lista med händelse innehållande information som plockats ut från tabellen.

Figur 6.5 visar den HTML-tabell som som innehåller kursnamn och kurskod. Informationen behandlas likt förgående operation, det vill säga med en XML-tolk. Kursnamnet kommer att användas längre fram som standardrubrik på alla händelser tillhörande schemat.

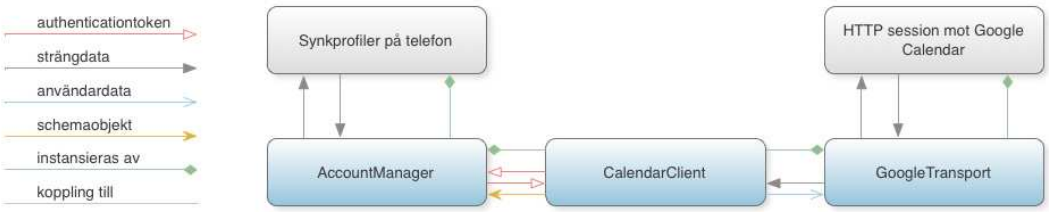
Processen som tolkar scheman körs i en egen tråd och med feedback till användaren med en grafisk komponent som indikator att schemat håller på att konverteras. På detta sätta avlastas gränssnittstråden från arbetet med konverteringen.

6.3.3 Användarinställningar

Användaren har vid start möjllghet att välja ett Google-konto som ska användas vid kommunikation med tjänsten Google Calendar. Kontona som visas ingår i den integrerade synkroniseringsfunktionen som är inbyggd i Android. De konton som går att lägga till i den funktionen avgöras av vilka program som är installerade de på enheten och om de har den möjligheten att synkroniseras med en extern tjänst. När användaren har valt ett konto så kommer endast namnet i synkroniseringsprofilen att användas i applikationen. Genom att senare begära tillstånd att läsa en authoriseringshash som genereras av synkroniseringsfunktionen kan användaren verifieras som behörig vid kommunikation med Google Calendar.

6.3.4 Skicka till Google Calendar

Kalenderklientens uppgift är att kommunicera med tjänsten Google Calendar. Uppgiften som kalenderklienten har är att hämta alla kalendrar och lägga in nya händelser/möten i en specificerad kalender. En användare som har ett konto hos Google Calendar har tillgång till flera olika kalendrar, vissa kalendrar äger användare själv eller så delas en kalender med en annan användare. Klienten som hämtar ner valbara kalendrar begränsar utbudet till de kalendrar som användaren själv skapat och därmed även har fulla rättigheter till. Alla egna kalendrar hämtar klienten ner sen bygger den upp objekt utifrån Xml-dokumentet den fick som svar på begäran med all information som finns om kalendrarna.



Figur 6.12: Kommunicerar med Google Calendar.

Klienten har nu tillgång till kalendrarna i ett hanterbart format och presenterar alla med respektive namn i en dialog-ruta där användaren kan välja kalender. För att skicka upp händelser till Google Calendar används ett HTTP-POST-meddelande innehållande ett XML-dokument som GoogleTransport har genererat.

Kapitel 7

Avslutande kommentarer

I detta kapitel kommer vi bland annat att utvärdera utvecklingsarbetet och de verktyg vi använt i framtagandet av vår applikation. Detta innefattar även Google Data API och de certifikat vi behövt för att testa applikationen på våra mobiltelefoner. Vi kommer även tala om de visioner vi har samt och de förbättringsåtgärder vi kan applicera på applikationen och slutligen kort gå igenom vad vi tyckte om projektet i sin helhet och vad vi lärt oss under tiden.

7.1 Utvärdering av utvecklingsmiljöerna

Utvecklingen av vår applikation har till stor del kretsat kring utvecklingsmiljöerna. Givetvis skulle man kunna skriva rubbet på en dammig Linux-burk i en simpel texteditor, men arbetet förenklas avsevärt när man har tillgång till bra verktyg. I detta avsnitt kommer vi att utvärdera miljöerna till iPhone och Andorid men även ta upp allt från klassreferenser och Google Data APIs till de certifikat vi använt.

7.1.1 iPhone

Själva installationen av iPhone SDK är inte direkt några konstigheter. Man skapar ett utvecklarkonto hos Apple, laddar ner programvaran, trycker install sen är det klart. Efter det man startat XCode, skapat ett nytt projekt och tryckt ”build and run” så kompileras applikationen och installeras i iPhone-simulatorn som i sin tur startar applikationen. Lätt som en plätt och utan komplikationer. Men för att kunna lägga in applikationen på själva enheten krävdes andra metoder. Man kan göra en så kallad ”jailbreak”, då man ges full access till telefonen och är fri att göra vad som helst i telefonens filsystem. Detta är dock inte rekommenderat och strider mot Apples riktlinjer. Den metod som vi valt att använda för att installera vår applikation innebar att skaffa sig ett Apple Developer Certificate, vilket kostar ca 1000 kronor per år. Med ett sådant har man i XCode installera och testa sin applikation direkt i enheten, samt lägga upp applikationen på applikationstorget App Store.

XCode har fungerat mycket bra. Det är lätt att felsöka, omfaktorisera och analysera för möjliga fel. När det gäller Interface Builder finns det fortfarande väldigt mycket kvar att lära sig, framförallt om hur man kombinerar kod och GUI. Programmet i sig är inte helt intuitivt vissa gånger. Något vi ännu inte har full koll på är exempelvis hur man kombinerar olika objekt med varandra och kopplar dessa till data och kontrollers.

Oavsett om det gäller verktygen eller språken så finns det väldigt mycket och bra information, dokumentation och guider om hur man utvecklar mot iPhone och Apple. Framförallt är Apples Developer Center en mycket bra källa till information.

Google Calendar API för Objective-C är något som i sig har fungerat väldigt bra. I deras API har de lagt in en hel del exempel på hur man kan använda sig av de olika komponenterna vilket gör funktionaliteten lätt att implementera. Däremot finns det nästan ingen dokumentation överhuvudtaget och det är otroligt svårt att få svar på ett problem som inte är beskrivet i ett exempel.

När det gäller Objektive-c var det förhållandevis lätt att komma in i hur det fungerar

och efter att ha använt det under en tid så tycker vi att det är ett bra språk. Framförallt är meddelande-systemet betydligt bekvämare än vad man till en början kan tänka sig. Genom att hela tiden veta vad det man stoppar in i en metods parametrar är för något gör koden väldigt självdokumenterande och enkel att läsa.

7.1.2 Android

Utvecklingsmiljön är lätt att komma igång med och genom officiella guider på Androids hemsida kan man vara igång för att kunna skriva en enkel *Hello world* applikation. Miljön erbjuder verktyg för att bygga gränsnitt vilket har varit till hjälp i separationen av gränssnittet och de lägre skikten i applikationen. Detta hindrar dock inte programmeraren från att kunna lägga in logik i vyn även om det är rekommenderat att hålla det så separerat som möjligt för att inte krångla till förändringsprocesser, utbyggnad och felsökning.

Simulatorn som ingår i SDK har all funktionalitet och att ha en egen Android-telefon för att utveckla applikationer är absolut inget måste. Simulatorn funkar precis som en vanlig telefon, applikationer går inte snabbare för att de körs i en simulator fastän datorkraften kan vara fem gånger så stor. För att testköra applikationen går det i simulatorn att genomföra scenario som till exempel att dataanslutningen inte funkar korrekt, samtal bryter programkörningen eller att orienteringen byts på skärmvyn till liggande. Däremot så finns inte Googles synkroniseringsprofiler tillgängliga i simulatorn. Detta har inneburit att vi inte kunnat testköra applikationen i simulatorn under det sista skedet av utvecklingen.

Google Calendar API till Andorid är än så länge i ett alphastadie vilket innebär att uppdateringsfrekvensen är väldigt hög. Under projektets gång har det släppts ett antal nya versioner som inte har hålls sig efter de tidigare riktlinjerna. Detta har resulterat i att vi tvingats arbeta om vissa delar av implementationen vid ett flertal tillfällen.

7.2 Neverlost Calendar

Implementationen av den idé och design vi från början hade har helt klart mött våra förväntningar. Vi fick det vi ville ha, det vill säga en liten och enkel applikation som gör att vi inte längre behöver surfa in, klicka runt och vänta på att webbplatsen ska uppdateras innan vi får veta när och var vi ska gå på en föreläsning. Det har helt enkelt blivit smidigare att hålla ordning på vilka dagar en kurs har föreläsning, vilken sal man ska gå till och vilken tid den börjar. Lösningen i sig är så pass bra att vi kan tänka oss, inom tid att sprida den till allmänheten. Men före detta sker finns det saker vi skulle vilja förändra. För även om kraven är nådda så finns det förbättringsåtgärder, eller kanske man vill kalla det för visioner. Under nästa avsnitt kommer vi gå igenom några av de förbättringsåtgärder som vi skulle vilja genomföra.

7.2.1 Förbättringsåtgärder och visioner

Under projektet med applikationen vi lyckats att implementera de krav vi satte i början av projektet. Men trots detta finns det förbättringsåtgärder. I detta avsnitt kommer vi gå igenom några av de förbättringsåtgärder som vi skulle vilja genomföra.

Först och främst finns det ett antal förbättringar som inte har implementerats på grund av tidsbrist, bland annat stödet för att ställa in påminnelser. Just nu så har alla händelser en påminnelse 10 minuter före händelsens starttid, något som man kanske vill ändra på eller helt ta bort. För att detta skulle vara användbart bör man även kunna redigera enskilda händelsers information, allt från titel till datum (om så vore att datumet till exempel inte stämmer).

Historik av redan nedladdade scheman är en annan sak som vore trevligt att ha i applikationen. Användarnyttan vore att kunna jämföra ett schema man redan lagt in med det schemat som just nu finns tillgängligt via Neverlost. Eftersom Neverlost inte tillhandahåller något sätt att identifiera en post så blir synkronisering av olika scheman omöjlig. Men om

användaren skulle kunna göra en egen tolkning av vad som är ändrat och inte kan man nog med enkla medel göra en synkronisering relativt enkelt. Det bäste vore om man skulle få tillgång att använda Neverlosts databas för synkronisering.

För att applikationen ska inrikta sig mot en så stor användargrupp som möjligt har vi valt att all tex är skriven på engelska. Det vore däremot kul om applikationen hade stöd för att användas sig av telefonens språkinställningar. För om man då har inställt på svenska kommer applikationen att ladda en svensk språkfil eller en engelsk.

Problemet med att redaktörerna av Neverlost lägger in lokalnamn under moment är något som går att fixa. Det man kan göra är att implementera en strängsökningsalgoritm som jämför moment mot en lista med kända lokalnamn. Om ett lokalnamn då påträffas kan strängen flyttas direkt till händelsens fält för lokal. Vi har även lekt med tanken att man skulle skapa en karta över Karlstad universitet och på den kartan skriva ut alla lokaler. För om student inte vet var den ska gå ska han/hon kunna trycka på händelsens fält för lokal och skickas vidare till kartan lokalens position.

Vi skulle även vilja implementera funktioner för kalenderkonfiguration. Med detta menar vi att man skulle kunna lägga in kalendern endast lokalt på telefonen genom Neverlost Calendar. Eftersom den lokala kalendern i sig kan vara kopplad till en serverkalender skulle vi utöka användarnyttan och inte låsa användaren till endast Googles kalendertjänst. Det vore även bra om man hade möjligheten att lägga till egna kalender-konton genom applikationen men även stöd för att skapa en helt ny kalender.

Något som har varit uppe för debatt redan innan vi börjande projektet var problemet med registreringskod kontra kurskod. Registreringskoden är endast aktuell före man börjar kursen, kurskoden å andra sidan är den beteckning som studenten under kursens gång kommer i kontakt med. Vi skulle därför vilja implementera stöd för att använda kurskoden istället för registreringskoden men vi har även tänkt ett steg längre. Om man i stället för att skriva in en kod fick bläddra bland alla program och kurser som går skulle man helt och hållet kunna ta bort problematiken med koder.

7.2.2 Slutord

Arbetet med att ta fram applikationen har varit otroligt lärorikt och intressant. Det känns framförallt kul att få arbeta med den nyaste och hetaste tekniken och göra något som inte bara vi, men andra kan ha nytta av. Den stora fördelen med att utveckla mot dessa plattformar i jämförelse med att utveckla mot Microsoft Windows eller Mac OS X är att många etablerade koncept ännu inte är utvecklade. Men även om det redan existerar en lösning så finns det fortfarande utrymme att göra egna tolkningar och förbättringar. Det behöver heller inte betyda år av kontinuerlig utveckling av enorma system, utan endast en smart liten applikation som gör livet lite bättre.

En annan sak vi reflekterat över är hur annorlunda det är att vara så begränsad. Det handlar inte bara om resurser i form av minne och processorhastighet men även skärmyta, komponenter och bibliotek. Men även om man är begränsad i vissa avseenden känns möjligheterna oändliga. Att till exempel istället för mus och tangentbord kunna röra vid det som händer på skärmen, att kunna styra och förändra det man ser innebär helt nya förutsättningar och kräver nytänkande och kreativitet.

Än så länge har vi endast fått ett smakprov på hur det är att utveckla till dessa plattformar. Vi har förvisso lärt oss hur man kommer igång, var potentialen ligger och fått känna hur långt ens fantasi kan sträcka sig. Men det finns otroligt mycket kvar att lära sig och vi tycker båda att det vore otroligt kul att i framtiden få arbeta med liknande teknik.

Referenser

[1] Androlib. Androlib statistics. http://www.androlib.com/appstats.aspx, Maj 2010. [Senast besökt 17-Maj-2010].

[2] Apache. Apache license 2.0. http://www.apache.org/licenses/LICENSE-2.0.html, September 2008. [Senast besökt 17-Maj-2010].

[3] Apple. Introduction to apple human interface guidelines. http://developer.apple.com/mac/library/documentation/UserExperience/Conceptual/AppleHIGuidelines/XHIGIntro/XHIGIntro.html, Augusti 2009. [Senast besökt 20-Maj-2010].

[4] Apple. Cocoa. http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html, Februari 2010. [Senast besökt 17-Mars-2010].

[5] Gareth Beavis. A complete history of android. http://www.techradar.com/news/phone-and-communications/mobile-phones/a-complete-history-of-android-470327, September 2008. [Senast besökt 17-Mars-2010].

[6] Henrik Bäck. Neverlost 2 icalendar — converter. http://it4.baeck.se/neverlost2ics/, 2010. [Senast besökt 24-Maj-2010].

[7] Mark Dalrymple and Scott Knaster. *Learn Objective-C on the Mac.* Apress, 1st edition, 2009. 9781430218159.

[8] Dropbox. Dropbox features. http://www.dropbox.com/features, 2010. [Senast besökt 19-Maj-2010].

[9] R. Fielding, UC Irvine, J. Gettys, Compaq/W3C, J. Mogul, Compaq, H. Frystyk, W3C/MIT, L. Masinter, Xerox, P. Leach, Microsoft, T. Berners-Lee, and W3C/MIT. HTTP RFC 2616 — ietf. http://tools.ietf.org/html/rfc2616, June 1999. [Senast besökt 24-Maj-2010].

[10] Google. Google APIs Maps External Library. `http://code.google.com/intl/sv-SE/android/add-ons/google-apis/`, June 2010. [Senast besökt 24-Maj-2010].

[11] Google. Google Calendar API. `http://code.google.com/apis/calendar/data/2.0/developers_guide.html`, 2010. [Senast besökt 29-Mars-2010].

[12] Ivor Horton. *Beginning C: from novice to professional*. Apress, 4th edition, 2006. 9781590597354.

[13] HTC. Htc Europe. `http://www.htc.com/europe/`, September 2008. [Senast besökt 15-April-2010].

[14] Rasmus Lerdorf, Kevin Tatroe, and Peter MacIntyre. *Programming PHP*. O'Reilly Media, Inc., 2nd edition, 2006. 9780596006815.

[15] Yukihiro Matsumoto and David Flanagan. *The Ruby Programming Language*. O'Reilly, 1st edition, Maj 2008. 9780596516178.

[16] Per-Anders Månsson. Neverlost. `http://neverlost.hb.se/portalen/portal.php?moduleIdentifier=cms&eventType=interface&eventName=introduktion`, November 2004. [Senast besökt 29-Mars-2010].

[17] OHA. Android 1.5 higlights. `http://developer.android.com/sdk/android-1.5-highlights.html`, April 2009. [Senast besökt 17-Mars-2010].

[18] OHA. Android 1.6 higlights. `http://developer.android.com/sdk/android-1.6-highlights.html`, September 2009. [Senast besökt 17-Mars-2010].

[19] OHA. Android 2.0 higlights. `http://developer.android.com/sdk/android-2.0-highlights.html`, Oktober 2009. [Senast besökt 17-Mars-2010].

[20] OHA. Open Handset Alliance. `http://www.openhandsetalliance.com`, Januari 2010. [Senast besökt 28-Mars-2010].

[21] OHA. User interface guidelines. `http://developer.android.com/guide/practices/ui_guidelines/index.html`, Maj 2010. [Senast besökt 20-Maj-2010].

[22] Richard. Jneverlost. `http://bluefire.dnsalias.com/~richard/jneverlost/`, 2007. [Senast besökt 24-Maj-2010].

[23] Rick Rogers, John Lombardo, Zigurd Mednicks, and Blake Meike. *Android Application Development*. O'Reilly, 1st edition, Maj 2009. 978059652147.

[24] Herbert Schildt. *Java: a beginner's guide*. McGraw-Hill Professional, 4th edition, 2006. 9780072263848.

[25] Android Team. Icon design guidelines, Android 2.0. http://developer.android.com/guide/practices/ui_guidelines/icon_design.html, April 2010. [Senast besökt 3-Maj-2010].

[26] W3C. Html. http://www.w3.org/TR/html401/, December 1999. [Senast besökt 28-Mars-2010].

[27] W3C. Extensible markup language (xml) 1.0 (fifth edition). http://www.w3.org/TR/xml/, November 2008. [Senast besökt 24-Maj-2010].

[28] W3C. HTML5. http://www.w3.org/TR/html5/, Januari 2010. [Senast besökt 28-Mars-2010].

[29] Wikipedia. App Store — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/App_Store, Februari 2010. [Senast besökt 17-Mars-2010].

[30] Wikipedia. Apple Inc. — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Apple_inc, Februari 2010. [Senast besökt 17-Mars-2010].

[31] Wikipedia. Application Programming Interface — Wikipedia, den fria encyklopedin. http://sv.wikipedia.org/wiki/API, 2010. [Senast besökt 20-Maj-2010].

[32] Wikipedia. Dalvik — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Dalvik_(software), Februari 2010. [Senast besökt 17-Mars-2010].

[33] Wikipedia. Designmönster — Wikipedia, den fria encyklopedin. http://sv.wikipedia.org/wiki/Designmœnster, 2010. [Senast besökt 20-Maj-2010].

[34] Wikipedia. Eclipse — Wikipedia, den fria encyklopedin. http://sv.wikipedia.org/wiki/Eclipse, Januari 2010. [Senast besökt 17-Februari-2010.

[35] Wikipedia. HTTP Request methods — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods, 2010. [Senast besökt 22-Maj-2010].

[36] Wikipedia. iPhone — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/IPhone, Februari 2010. [Senast besökt 17-Mars-2010].

[37] Wikipedia. iPhone SDK — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/IPhone_OS\#iPhone_SDK, Maj 2010. [Senast besökt 29-Mars-2010].

[38] Wikipedia. iPod — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Ipod, Februari 2010. [Senast besökt 17-Mars-2010].

[39] Wikipedia. iTunes — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/ITunes, Februari 2010. [Senast besökt 17-Mars-2010].

[40] Wikipedia. iTunes store — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/ITunes_Store, Februari 2010. [Senast besökt 17-Mars-2010].

[41] Wikipedia. Model-view-controller — Wikipedia, den fria encyklopedin. http://sv.wikipedia.org/wiki/Model-View-Controller, 2010. [Senast besökt 20-Maj-2010].

[42] Wikipedia. Object-oriented programming — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Object-oriented_programming, Februari 2010. [Senast besökt 17-Mars-2010].

[43] Wikipedia. Objective-C — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Objective-c, Februari 2010. [Senast besökt 17-Mars-2010].

[44] Wikipedia. Smalltalk — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Smalltalk, Februari 2010. [Senast besökt 17-Mars-2010].

[45] Wikipedia. Uniform Resource Locator — Wikipedia, den fria encyklopedin. http://sv.wikipedia.org/wiki/Uniform_Resource_Locator, April 2010. [Senast besökt 22-Maj-2010].