Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Neža Čeč

Načrtovanje poslovnih aplikacij za zaslon na dotik

diplomsko delo

univerzitetni študijski program računalništvo in informatika

Mentor: izr. prof. dr. Viljan Mahnič

Ljubljana, 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01896/2013
Datum: 01.02.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **NEŽA ČEČ**

Naslov: **NAČRTOVANJE POSLOVNIH APLIKACIJ ZA ZASLON NA DOTIK**
DESIGNING TOUCHSCREEN BUSINESS APPLICATIONS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Predstavite značilnosti aplikacij, ki za interakcijo z uporabnikom uporabljajo zaslon na dotik. Opišite posebnosti, ki jih je treba upoštevati pri načrtovanju uporabniškega vmesnika in izbiri ustrezne tehnološke rešitve. Na podlagi tega predelajte eno izmed obstoječih poslovnih aplikacij v mobilno aplikacijo, ki bo uporabljala zaslon na dotik. Podrobno predstavite načrtovanje izgleda, izbrano tehnologijo in sam postopek razvoja. Na koncu primerjajte med sabo obe rešitvi.

Mentor:
prof. dr. Viljan Mahnič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Neža Čeč, z vpisno številko **63080030**, sem avtor diplomskega dela z naslovom:

Načrtovanje poslovnih aplikacij za zaslon na dotik

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom izr. prof. dr. Viljana Mahniča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne Podpis avtorja:

Iskreno se zahvaljujem svojemu mentorju, prof. dr. Viljanu Mahniču, za hitre in izčrpne odgovore ter drugo pomoč pri izdelavi diplomskega dela.

Zahvaljujem se tudi Petru Brajaku za ideje, nasvete in ker mi je dal možnost pisati diplomo v podjetju Medius.

Za potrpežljivo pomoč in za odgovore na mojo neskončno vrsto vprašanj pri razvoju aplikacije se zahvaljujem Martinu Bolčini.

Nenazadnje se zahvaljujem svoji družini za podporo.

Posebna zahvala pa gre tudi Joštu Žerjavu, ki je enkrat omenil, da ga ne smem izpustiti v zahvali. Hvala tudi tebi, dragi, za vse.

Kazalo

Povzetek

Diplomska naloga opisuje prepis poslovne aplikacije, namenjene za tipkovnico in miško v aplikacijo za zaslon na dotik. Najprej so opisane funkcionalne značilnosti aplikacij za zaslon na dotik, kot sta naprimer njihova zgradba in interakcija z njimi. Nato so opisane tehnološke značilnosti, ki pokrivajo tehnologije in ogrodja, ki jih imamo na voljo za razvoj. Sledi praktični primer prepisa aplikacije v obliko za zaslon na dotik, ki je sestavljen iz načrtovanja izgleda in podrobnega opisa razvoja. Sledi še primerjava obeh vrst aplikacij, analiza razvoja, končnega izdelka, ter kriterijev za nadaljni razvoj poslovnih aplikacij za zaslon na dotik.

Ključne besede

poslovna aplikacija, zaslon na dotik, načrtovanje aplikacije, razvoj aplikacije

Abstract

This thesis covers conversion of a business application from a standard application to a touch screen version. The first part describes basic functional characteristics of touch screen applications, like their structure and user interaction. Later on we describe technological characteristics that cover different technologies and frameworks for developing touch screen applications. These theoretic chapters are followed by a practical example of converting a business application to a touch screen version. The example includes design planning and detailed development description. In the end we compare both types of application and make an analysis of development, final product and criteria for developing a business touch screen application.

Keywords

business application, touch screen, designing application, application development

Poglavje 1

Uvod

V zadnjih letih lahko opazimo, da se je uporabnost in priljubljenost zaslonov na dotik močno povečala. Večina ljudi namreč že uporablja tako imenovane pametne telefone s takim zaslonom, marsikdo pa je tudi že lastnik tabličnega računalnika, ki mu nadomešča prenosni računalnik. Na porast priljubljenosti teh naprav kažejo različne analize, naprimer kitajska raziskava, ki je pokazala, da več ljudi dostopa do interneta preko pametnih telefonov, kot preko običajnih računalnikov [10].

Zaradi priljubljenosti zaslonov na dotik na mobilnih napravah pa se poraja vprašanje, kdaj se bodo zasloni na dotik pojavili na prenosnih in stacionarnih računalnikih. Strokovnjaki si sicer glede tega niso enotni, a nam zelo dobro predstavo o prihodnosti zaslonov na dotik ponuja zadnja poteza podjetja Microsoft. Večina uporabnikov namreč uporablja eno od verzij operacijskih sistemov Windows (npr. Windows XP, Windows Vista ali Windows 7), po nekaterih podatkih tudi več kot 70%. Konec oktobra 2012 je podjetje Microsoft dalo na trg nov operacijski sistem Windows 8, ki se od predhodnih različic razlikuje najbolj doslej, razlog za to pa je ravno dejstvo, da je Windows 8 prilagojen predvsem za zaslone na dotik. V prihodnosti bo Windows 8 edini Microsoftov operacijski sistem, ki bo na voljo za osebne računalnike, hkrati pa bo za tablične računalnike na voljo Windows 8 RT, za pametne telefone pa Windows Phone 8. To je izredno pomembno dejstvo, saj naka-

zuje, da bomo v prihodnosti na vseh napravah imeli zaslone na dotik, kar predstavlja vsaj za razvijalce namiznih in spletnih aplikacij precejšnjo spremembo. Za trenutek se želim še ustaviti pri razlogih, ki so Microsoft vodili k razvoju takega operacijskega sistema. Pri The Economist-u predvidevajo, da se prodaja računalnikov z zasloni na dotik ne bo resnično razširila najmanj do leta 2014, čeprav pravijo, da se ne moremo ogniti dejstvu, da se trendi v računalništvu premikajo od uporabe tipkovnice in miške proti zaslonom na dotik. Iz tega lahko sklepamo, da razvoj dogodkov Microsoftu ni pustil druge možnosti, kot da se prilagodi trendom in izkoristi željo uporabnikov po enakem operacijskem sistemu na računalnikih in mobilnih napravah [9]. Poleg Microsofta je tudi konkurenčno podjetje Google po zadnjih podatkih napovedalo prenosnik z zaslonom na dotik, ki ga bodo na trg poslali verjetno že letos, tretji konkurent, podjetje Apple, pa že dolgo kraljuje na področju tehnologije zaslonov na dotik.

Sedaj pa se lahko vprašamo, kaj to pomeni za razvijalce različnih poslovnih aplikacij. Trenutno so bolj ali manj vse poslovne aplikacije narejene za uporabo s tipkovnico in miško, le redke so na voljo tudi v verziji za mobilne naprave, oziroma za zaslon na dotik. Zaradi tega se moramo razvijalci poslovnih aplikacij zavedati, da vse naše znanje o izgledu in razvoju aplikacij temelji na predpostavki, da uporabnik uporablja tipkovnico in miško, ter ima na voljo sorazmerno velik zaslon. A vendar je jasno, da bodo uporabniki slej ko prej želeli svoje poslovne aplikacije uporabljati tudi na poti preko mobilnih naprav ali pa si pomagati z zaslonom na dotik, ko bo le ta v širši uporabi tudi pri običajnih računalnikih. Jasno je torej, da je potrebno tudi razvoj poslovnih aplikacij prilagoditi tem trendom, najbolje v čim krajšem času, da nas kasneje hitre spremembe trga ne presenetijo. Pri tem bo treba upoštevati veliko različnih lastnosti naprav in platform, na katerih bomo želeli uporabljati svoje poslovne aplikacije. Upoštevati bo potrebno lastnosti, kot je naprimer velikost zaslona in moč procesorja, druga novost pa je načrtovanje aplikacij specifično za zaslon na dotik. To predvsem za poslovne aplikacije pomeni ogromno spremembo, saj se morajo spremeniti tudi maske, oziroma osnovni

izgled aplikacij. Poleg tega smo trenutno v obdobju, ki bo zahtevalo prehod iz ene tehnologije v drugo, kar pomeni da bodo vmes morda morale aplikacije podpirati obe možnosti uporabe.

Na tem mestu je smiselno tudi omeniti terminologijo, ki jo želim uporabljati. Že v tem uvodu je bilo vidno, da izmenično uporabljam izraza aplikacija za zaslon na dotik in mobilna aplikacija. Pridevnik mobilno se nanaša na izraz mobilni telefon, vendar pa danes za večino novih telefonov uporabljamo nov izraz in sicer pametni telefon. Večina novih telefonov ima tudi zaslon na dotik in aplikacijam, ki tečejo na njih, pravimo kar mobilne aplikacije. Zato se po mojem mnenju ta dva pojma danes prepletata. To nakazuje tudi dejstvo, da med mobilne aplikacije štejemo tudi aplikacije za tablične računalnike, ki imajo seveda tudi vsi zaslone na dotik. Do večje zmede pa bo prišlo ravno s prihodom računalnikov z zaslonom na dotik, saj bodo na njih lahko enake aplikacije kot na tabličnih računalnikih in pametnih telefonih, vendar za te aplikacije običajno ne bi uporabili izraza mobilna aplikacija. Zato si želim, da bi obstajal boljši izraz za to, vendar bom v nadaljevanju uporabljala izraza mobilna aplikacija in aplikacija za zaslon na dotik enakovredno. Razlog za to je tudi dejstvo, da naj bi bile te aplikacije resnično mobilne, vendar v drugem pomenu: mobilne med različnimi platformami.

Sledeče besedilo je sestavljeno iz dveh delov. Prvi del predstavlja teoretično podlago razvoja aplikacij za zaslone na dotik ter temelji predvsem na literaturi, ki je trenutno na voljo o aplikacijah za zaslone na dotik. Ker so te aplikacije danes omejene večinoma na mobilne telefone, se tudi večina primerov nanaša nanje. Sicer pa je prvi del sestavljen iz dveh poglavij. Prvo se nanaša na samo funkcionalnost aplikacij za zaslon na dotik in uporabnikovo interakcijo s temi aplikacijami, drugo pa na tehnološki vidik in ogrodja, ki jih lahko uporabimo za razvoj takih aplikacij. Drugi del pa je praktični primer prepisa obstoječe aplikacije v mobilno aplikacijo in analiza končnega izdelka. Rezultat obeh delov je dobra osnova za načrtovanje razvoja poslovne aplikacije za zaslon na dotik, poleg tega pa dobimo tudi idejo o tem, kdaj se je takega podviga smiselno lotiti in kdaj ne.

Poglavje 2

Funkcionalne značilnosti aplikacij za zaslon na dotik

2.1 Uvod

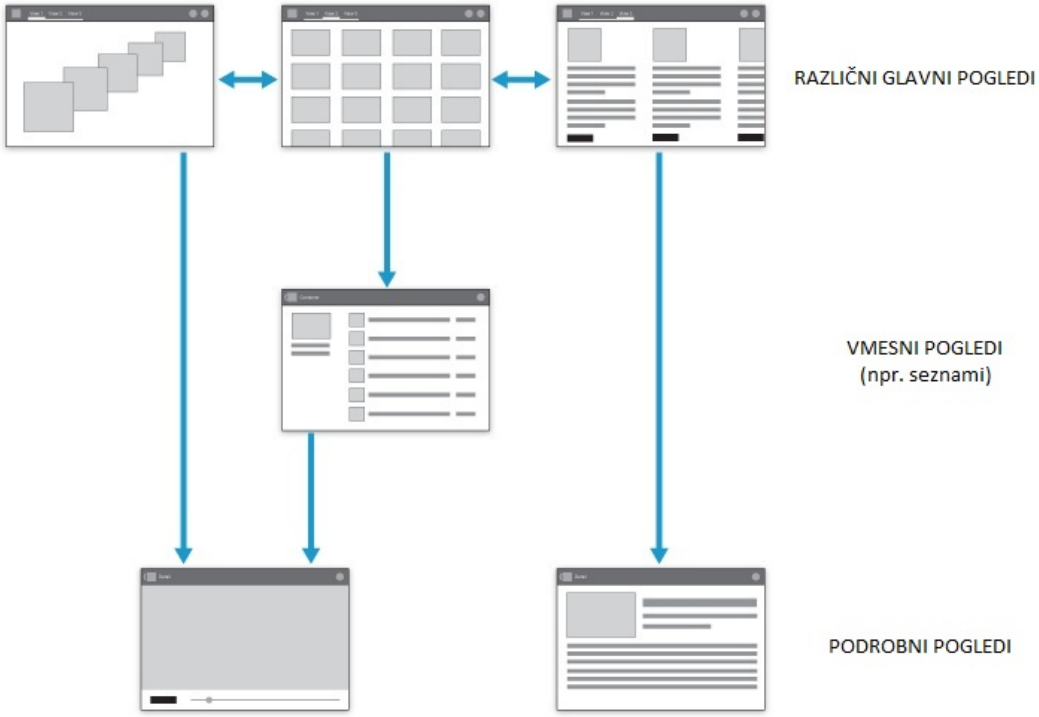
To poglavje obsega pregled zgradbe in izgleda aplikacij za zaslon na dotik, uporabnikove interakcije z aplikacijo in pristope ter pravila, ki se jih moramo držati pri razvoju. Gre predvsem za pregled osnovnih konceptov, ki nam otežujejo načrtovanje take aplikacije, ki pa veljajo tako za poslovne in druge aplikacije. Pri poslovnih aplikacijah je morda naloga celo težja, saj so običajno namenjene pregledovanju, urejanju, dodajanju in brisanju različnih vnosov, poleg tega pa nam pogosto omogočajo tudi opravljanje statistik, pripravo poročil in izvajanje mnogo drugih akcij. To pomeni da so lahko precej obširne in jih je zato težje narediti pregledne in organizirane. Sestavljajo jih gradniki kot so meniji, tabele, grafi, seznami, gumbi, zavihki, forme, vnosna polja itd. Pogosto nam lahko prikažejo tudi veliko teksta ali podrobnosti o posameznem objektu, ki ga pregledujemo, kar je predvsem na manjših zaslonih lahko izziv. Cilj je torej oblikovati uporabniku prijazne in intuitivne aplikacije, z upoštevanjem vseh omejitev in izkoriščanjem vseh prednosti zaslona na dotik, ne da bi ob tem izgubili katero od funkcij aplikacije.

2.2 Izgled in zgradba aplikacije

Pri razvoju aplikacij za običajne računalnike lahko predvidevamo, da je uporabnikov zaslon dokaj velik in da za izbiro elementov na zaslonu uporablja miško, ki je v primerjavi s človeškimi prsti zelo natančno orodje. To sta dve od osnovnih težav pri načrtovanju zgradbe aplikacije za zaslon na dotik. Prva je torej velikost zaslona, ki je lahko pri telefonu zelo majhen, pri prenosniku pa velik. Če upoštevamo torej še, da so človeški prsti dokaj nenatančni in morajo biti zato gumbi na zaslonu večji kot za miško, je torej jasno da se moramo načrtovanja uporabniškega vmesnika za tako aplikacijo lotiti še posebej natančno. Še preden pa se lotimo načrtovanja aplikacije, pa bi bilo dobro osvetliti še eno vprašanje, ki se bo pojavljalo skozi celoten razvoj. Želimo si namreč, da bi naša poslovna aplikacija tekla na vseh operacijskih sistemih in napravah, hkrati pa želimo, da bi imeli z razvojem čim manj dela. Vprašanje torej je, koliko samo aplikacijo prilagoditi platformi, na kateri se izvaja. Kot bomo videli v nadaljevanju, so razlike med značilnostmi aplikacij na posameznih platformah lahko velike. Če se odločimo, da aplikacijo naredimo specifično glede na platformo, bomo za posamezen del potrebovali dalj časa in s tem zvišali stroške razvoja.

2.2.1 Gradniki

Podobno kot običajne aplikacije, tudi mobilne aplikacije sestavljajo enaki gradniki, samo uporabljati jih moramo čim bolj smiselno, da izkoristimo prostor na zaslonu. Ti gradniki so enaki oziroma podobni tudi na vseh platformah, zato lahko vzamemo primere z ene platforme in podobno velja tudi za preostale. Glede na zahteve in kompleksnost aplikacije, je odvisno, koliko različnih pogledov bomo potrebovali in kako globoka bo hierarhija strukture aplikacije. Kot je prikazano na sliki 2.1, imamo na najvišjem nivoju glavni pogled aplikacije, na srednjem nivoju imamo lahko neke sezname, oziroma bolj specifične poglede v primerjavi z glavnimi, na najnižjem pa prikaz podrobnosti elementa ali poglede za urejanje [14].

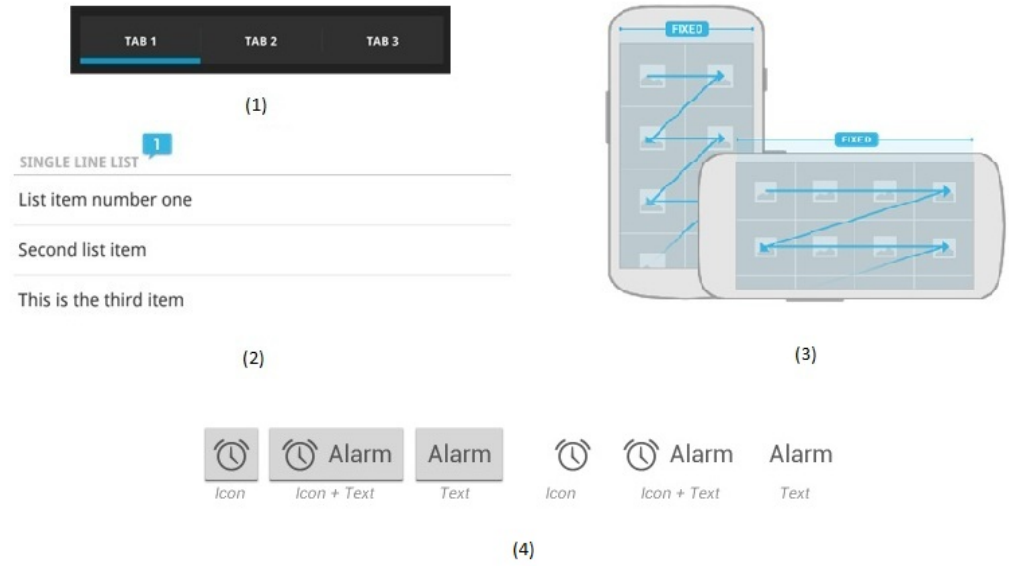


Slika 2.1: Hierarhija pogledov v aplikaciji

Organiziranost različnih pogledov je v mobilnih aplikacijah izrednega pomena, saj so te aplikacije namenjene za naprave z različno velikimi zasloni, različnimi resolucijami zaslonov in različno postavitvijo, pokončno ali ležečo. Kot bomo videli v nadaljevanju je torej pomembno, da se lahko ti pogledi različno prilagajajo. Med različnimi pogledi se sprehajamo z uporabo navigacije, ki je ponavadi kar v obliki gumbov, pri tem pa je pomembno, da uporabnik ne izgubi občutka za to, kje v aplikaciji se trenutno nahaja. Sicer pa poglede sestavljajo gradniki, kot so zavihki, seznami, mrežni seznami, drsniki, spustni meniji, gumbi, polja, obvestila in še mnogo drugih (slika 2.2).

2.2.2 Prilagajanje zaslonu

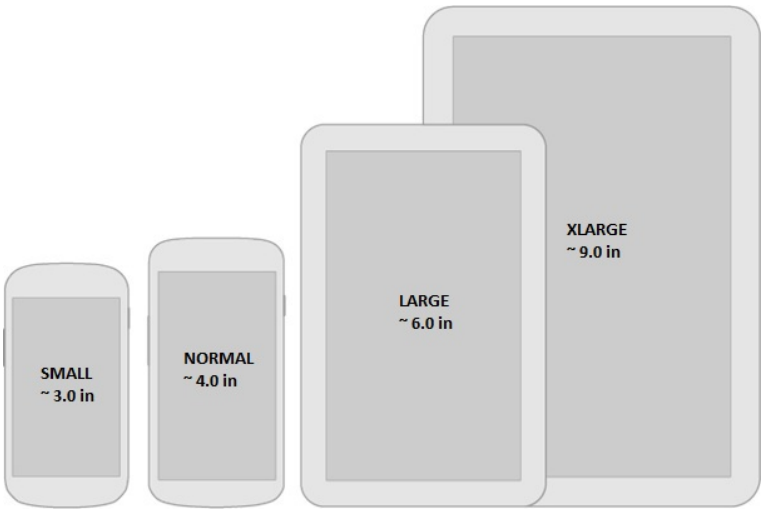
Kot je bilo že večkrat omenjeno, je pri mobilnih aplikacijah ena od večjih ovir velikost zaslona, njegova resolucija in postavitev. Zato moramo pri



Slika 2.2: (1)primer zavihkov, (2)primer seznama, (3)primer mrežnega seznama, (4)primeri gumbov

načrtovanju aplikacije misliti tudi na to, da bo le ta na koncu primerna za vse naprave. Seveda bi lahko naredili aplikacijo popolnoma neprilagodljivo, vendar bi se nam lahko zgodilo, da bi bila na manjših zaslonih zelo nepregledna in neberljiva, na večjih pa okorna in preveč preprosta. Na sliki 2.3 lahko vidimo ogromno razliko v velikosti zaslonov med različnimi napravami. Merila, ki opisujejo posamezen zaslon so njegova fizična velikost, gostota slikovnih pik (število slikovnih pik na enoto dolžine, angl. dots per inch - dpi) in postavitev.

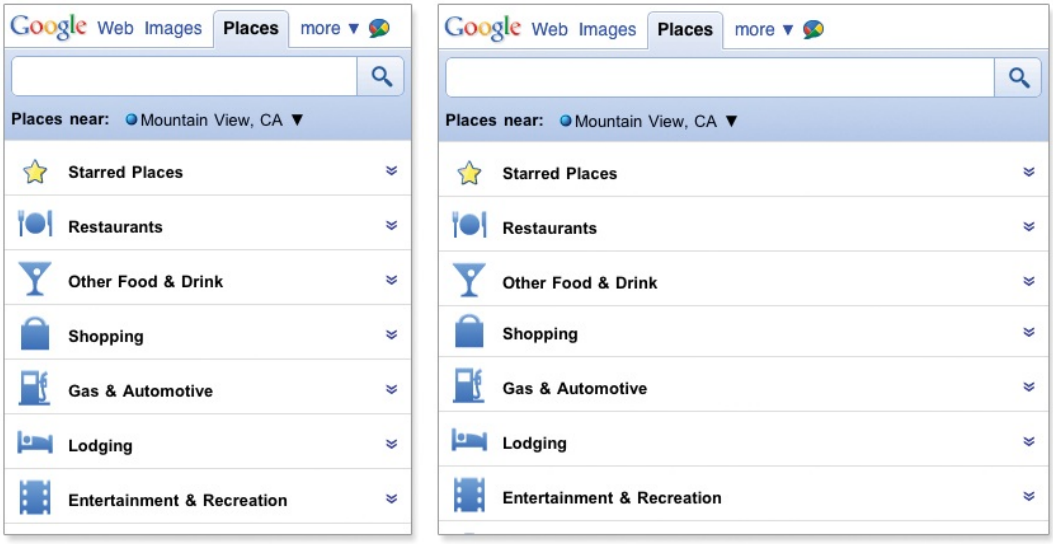
Na velikost zaslona se lahko prilagajamo z raztegovanjem ali krčenjem postavitve aplikacije, združevanjem različnih pogledov za boljši pregled in z različnimi velikostmi slik, ki jih prikazujemo. Raztegovanje ali krčenje postavitve pomeni, da svojo aplikacijo razvijemo za neko povprečno velikost zaslona, nato pa se postavitev avtomatsko prilagaja drugim širinam. Ta način je po svoje obvezen, saj le tako lahko dosežemo prilagajanje vsem vrstam zaslonov. Vendar pa ni dobro, da uporabljamo samo ta način prilagajanja.



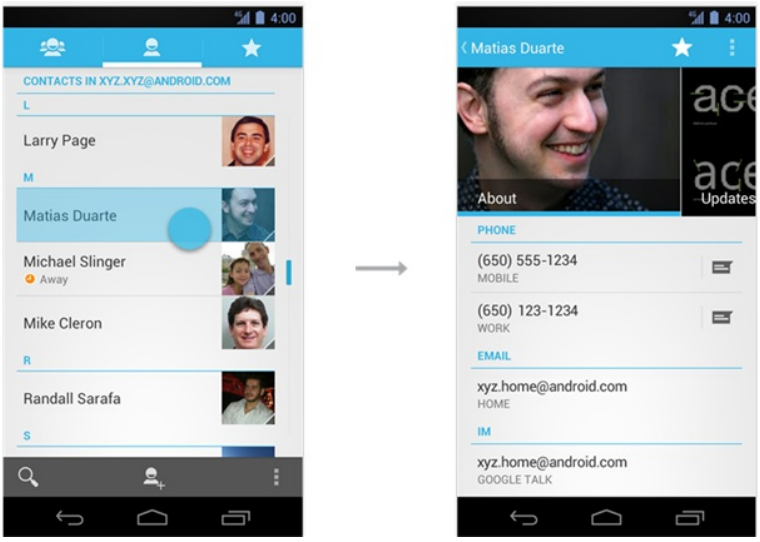
Slika 2.3: Primerjava velikosti zaslonov. Pri Androidu so zaslone uredili v različne velikostne skupine: majhni (2.0 in - 3.0 in), običajni (3.0 in - 5.0 in), veliki (5.0 in - 7.0 in) in zelo veliki (7.0 in - 10.0+ in)

Na večjih zaslonih se namreč lahko zgodi, da nam pogled raztegne celo na dvojno širino, kot je razvidno na sliki 2.4. To raztegovanje ne vpliva le na estetski izgled aplikacije, ampak tudi na uporabnost, saj z večjim zaslonom nismo pridobili ničesar, čeprav bi lahko razširili uporabnost aplikacije. Zato lahko v takih primerih uporabimo združevanje pogledov in dodajanje vsebine. To pomeni, da moramo naprimer na manjših zaslonih izbrati nek objekt s seznama, da se nam v drugem pogledu odprejo podrobnosti (slika 2.5), na večjem zaslonu pa lahko oba pogleda prikažemo hkrati (slika 2.6). Poleg tega lahko aplikacijo prilagajamo tudi z dodajanjem vsebin. Na manjših zaslonih prikažemo le osnovne podatke, na večjih pa bolj podrobne opise, slike in druge informacije (slika 2.7). Oba ta načina izkoriščata prostor kolikor se le da in tako povečata uporabnost, ter preglednost aplikacije. Vedeti moramo namreč, da je na manjših zaslonih bolje uporabniku ponuditi manj informacij in s tem ohraniti čistost uporabniškega vmesnika [2]. Zadnji način prilagajanja vsebine je, da aplikaciji damo na voljo več velikosti slik, ki jih lahko uporabi. To je pomembno zato, ker se naprave razlikujejo tudi v gostoti

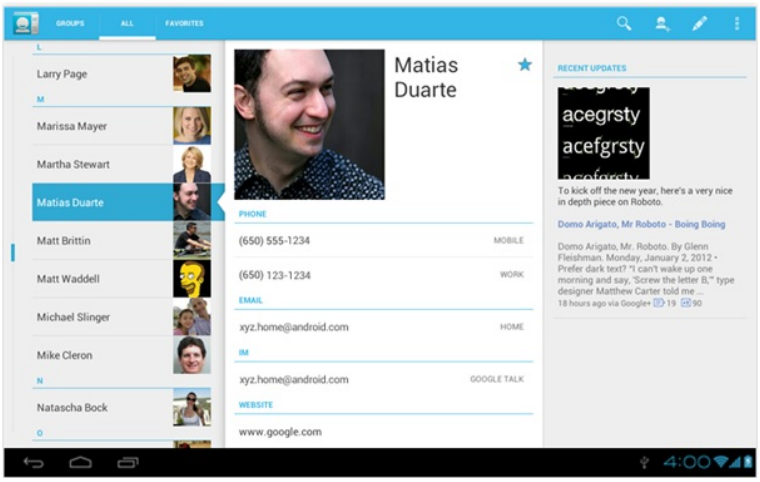
slikovnih pik. Če imamo na voljo le slike, ki so primerne za manjšo gostoto, se pri večji gostoti zgodi, da slika izgleda robato ali zabrisano, kot je razvidno na sliki 2.8. Poleg do sedaj naštetega prilagajanja, pa lahko posebej omenimo še menjavo med ležečo in pokončno postavitvijo naprave. Prilagodimo se lahko ponovno z raztegovanjem, reorganizacijo elementov, širjenjem posameznih podatkov ter skrivanjem in prikazovanjem posameznih elementov (slika 2.9).



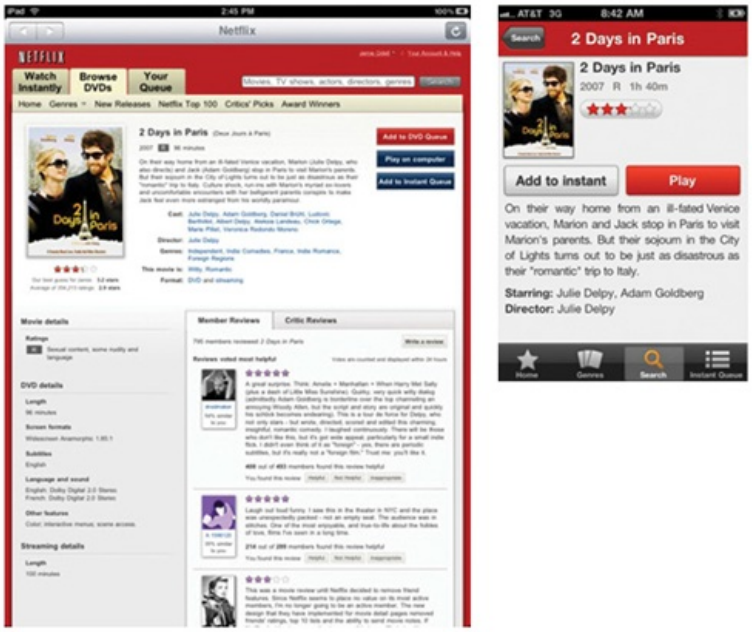
Slika 2.4: Primer raztega pogleda, s katerim dobimo veliko praznega prostora, ki bi ga lahko bolje izkoristili.



Slika 2.5: Na levi imamo pogled s seznamom, ob kliku na element seznama pa se nam odpre nov pogled s podrobnostmi.

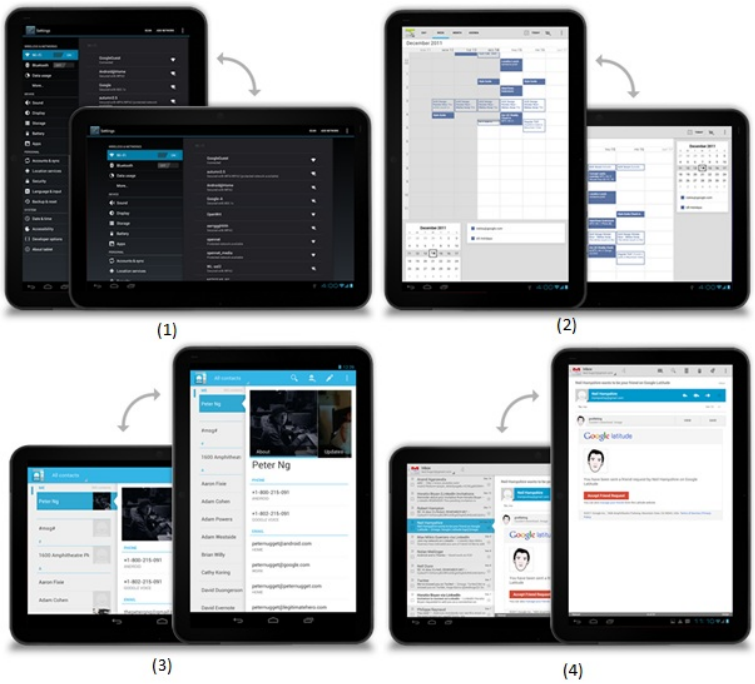


Slika 2.6: Pri večjih zaslonih lahko seznam in podrobnosti izbranega elementa seznama prikažemo v združenem pogledu.



Slika 2.7: Na večjih zaslonih lahko prikažemo več podrobnosti (primer na levi), kot na manjših (primer na desni).

Slika 2.8: Primer slabega prikaza slike na desni zaradi različnih gostot slikovnih pik.



Slika 2.9: (1)raztegovanje, (2)reorganizacija, (3)širjenje, (4)skrivanje posameznega elementa

2.2.3 Gumbi

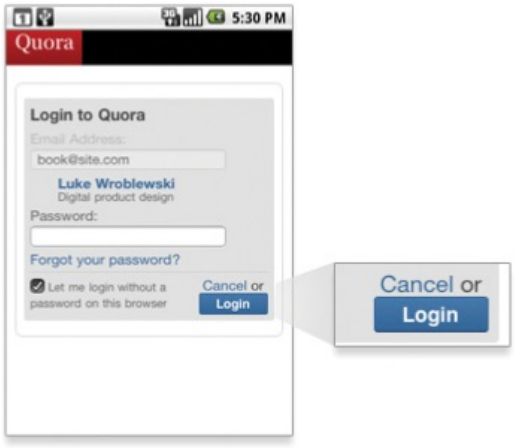
Gumbi so pri zaslonih na dotik eden od najbolj pomembnih elementov. Nanje pri uporabi z miško nismo bili več pozorni, pri uporabi prstov pa se stvari rahlo zapletejo. Glavna razlika med miško in prsti je natančnost. Z miško lahko zadenemo zelo majhne tarče, prsti pa so precej večji, razlikujejo se po velikosti od človeka do človeka in lahko na zaslonu tudi zdrsnejo ali kako drugače zgrešijo tarčo. Raziskava, ki so jo opravili pri Harris Interactive, je namreč pokazala, da je približno polovica klikov na reklame posledica napake [11]. Iz tega seveda sledi, da moramo v svoji aplikaciji povsod ponuditi možnost vrnitve, oziroma razveljavljenja akcij, vendar to ni edina rešitev. To možnost napake moramo kar se le da zmanjšati, da izboljšamo uporabniško izkušnjo. Zlato pravilo pri oblikovanju mobilnih aplikacij se torej glasi: večje je, boljše je. A velikost ni edina pomembna lastnost gumbov. Paziti moramo tudi pri razmaku in postavitvi na zaslonu.

Kako veliki morajo torej biti gumbi, da so do uporabnika čim bolj prijazni? Zaradi različnih velikosti slikovnih pik pri zaslonih, se za določanje velikosti priporoča uporaba fizičnih enot. Pri podjetju Apple zato priporočajo velikost gumba vsaj 44x44 pik, v navodilih za Windows Phone 7 pa priporočajo velikost 9mm, oziroma minimalno 7mm, ter 2mm vmesne razdalje. To sicer ne pomeni, da morajo biti vsi gumbi te velikosti, ampak mora celotno območje akcije ustrezati tem merilom. Slika, ki predstavlja gumb je seveda lahko manjša. To razliko lahko vidimo na sliki 2.10. Poleg tega lahko pri določanju velikosti gumba upoštevamo tudi preprosto pravilo – večkrat kot uporabljamo nek gumb, večji naj bo. Podobno lahko upoštevamo tudi pri postavitvi – blizu gumba z neko pomembno funkcijo, naj se ne nahaja kak drug gumb, saj s tem spet povečamo možnost napake (slika 2.11).

Sedaj ko smo že upoštevali velikost in razmak med gumbi, pa si lahko ogledamo še, zakaj je pomembna lokacija gumba na zaslonu. Misliti moramo namreč tudi na to, kako ljudje držijo telefon oziroma tablični računalnik v rokah in katere dele zaslona imajo na dosegu brez težav in kateri deli so bolj neprikladni. Zato gumbe za prekinitev ali brisanje raje postavimo v bolj

nedosegljive dele zaslona, da so čim bolj oddaljeni od drugih gumbov, hkrati pa se mora uporabnik bolj potruditi, da neko stvar izbriše. Na sliki 2.12 vidimo, kje so najboljše pozicije za gumbe, da jih uporabnik čim lažje doseže. Pri tem lahko poudarimo še, da na tiste dele, ki so s prsti bolj dosegljivi, dajemo manj teksta, saj jih večkrat prekrivamo z roko, kot pa oddaljene predele zaslona [16].

Slika 2.10: Razlika med dejansko in vidno tarčo



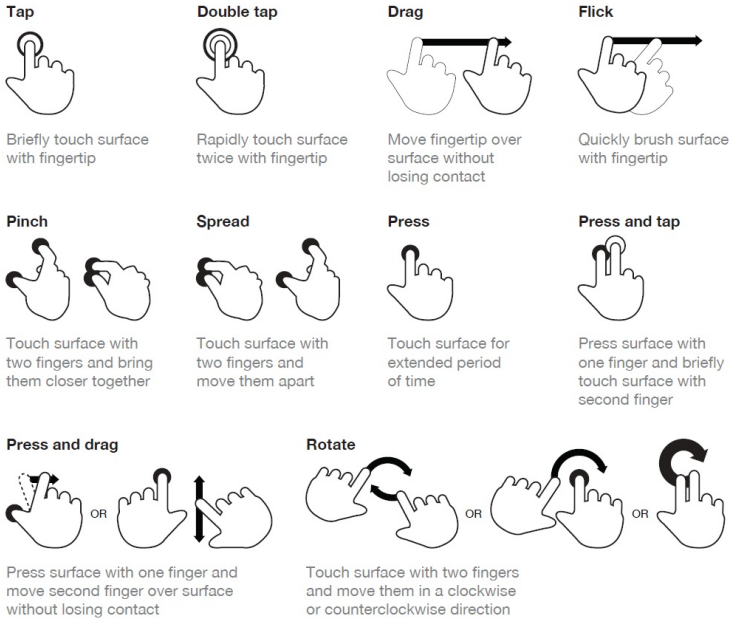
Slika 2.11: Primer slabe postavitve gumbov, kjer sta gumba Cancel in Login preblizu skupaj.

Slika 2.12: Boljše in slabše postavitve gumbov, glede na doseg palca.

2.3 Interakcija z aplikacijo

Interakcija z aplikacijo na zaslonih na dotik poteka na več načinov. V obliki dotikov imamo lahko osnovni enojni dotik, dotik z več prsti hkrati in različne kretnje, ki vključujejo razne potege, drsenje, primike in odmike (slika 2.13) [6].

Naštete kretnje so prisotne na večini platform, sicer pa obstajajo tudi kretnje, ki so prisotne samo na eni platformi, na drugih pa ne, kot je naprimer štiriprstni poteg v operacijskem sistemu iOS, ki služi za premik med aplikacijami. Pri poslovnih aplikacijah večinoma ne želimo preveč dodatnega dela, zato lahko take specifične akcije izpustimo in jih ne implementiramo, a težave predstavljajo tudi tisti gibi, ki so enaki na različnih platformah, vendar ne sprožijo enake akcije, kot je naprimer dolgi klik. Dolgi klik v Androidu sproži možnost izbire teksta, v Windowsih pa s tem gibom dobimo bolj podrobne informacije o elementu ali pa pomoč za uporabo. Zaradi tega tu naletimo na težavo ali se bomo prilagajali posamezni platformi, ali naredili delovanje na vseh enako. Argumenti seveda obstajajo za in proti, vendar za večino poslovnih aplikacij pride v poštev, da se ne spuščamo v specifike platforme, saj to zvišuje stroške razvoja, poleg tega pa med platformami vseeno obstaja dovolj podobnosti [2]. Eden od močnih argumentov je tudi dejstvo, da ljudje po vsem svetu za posamezne akcije želijo uporabljati enake kretnje, zaradi česar



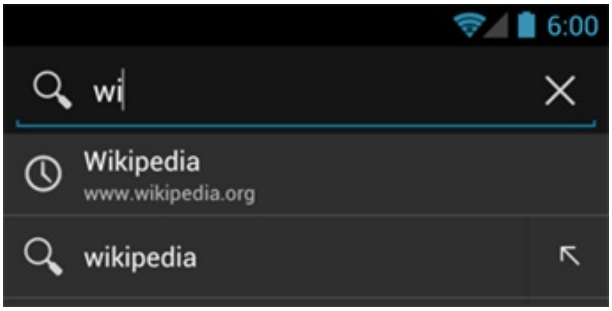
Slika 2.13: Pogoste kretnje, ki jih uporabljamo pri interakciji z aplikacijo.

lahko predvidevamo, da se bodo kretnje in akcije v prihodnosti poenotile [7]. To je opazno tudi zaradi dejstva, da najprej v eni aplikaciji uporabijo neko kretnjo, ki se ljudem zdi naravna in jo pogosto uporabljajo in tako se počasi razširi tudi v preostale aplikacije. Ena od takih kretenj je recimo poteg navzdol za osveževanje [2]. Zaradi te naravne direktne interakcije z aplikacijo, pravimo tudi, da je uporabniški vmesnik naraven (angl. Natural User Interface - NUI) za razliko od starega grafičnega vmesnika (angl. Graphical User Interface - GUI) [2]. Ker pa ti gibi za interakcijo niso vsem očitni, moramo v aplikacijo dodati tudi namige, ki se pojavijo prvič, ko imamo neko novo kretnjo na voljo. Tako uporabnika najlažje poučimo o hitri uporabi aplikacije in poskrbimo, da uporablja vse njene funkcije. Seveda pa moramo paziti, da uporabnika ne zasujemo z navodili in namigi, saj lahko postanejo nadležni, če jih je preveč.

Obstaja še ena razlika med miško in prsti. Pri uporabi miške se nekatere akcije sprožijo že, če miško samo postavimo na nek določen element na za-

slonu, ko miška t.i. lebdi nad elementom (angl. hover). Pri zaslonu na dotik pa nimamo te informacije, kdaj je prst nad nekim elementom, zato moramo akcije, ki delujejo na ta princip, za zaslon na dotik spremeniti. Vendar to ni večja ovira, saj je to zaznavanje lokacije miške le prikladnost, ne nujnost [2].

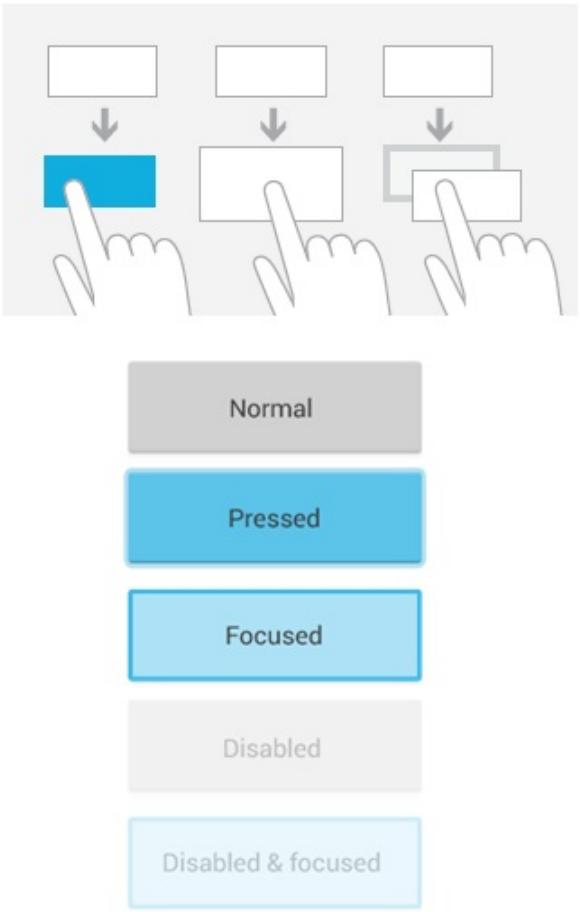
Nazadnje pa se lahko ustavimo še pri tipkovnici. Strokovnjaki predvidevajo, da bo tudi v prihodnosti tipkovnica ostala del komunikacije z napravo, vsaj pri računalnikih, saj ljudje neradi uporabljajo virtualne tipkovnice na zaslonih na dotik. To je seveda pomembno predvsem, kadar imamo aplikacijo, s katero želimo vnašati veliko teksta, kar je pogosto pri poslovnih aplikacijah. Vendar to ne predstavlja bistvene razlike, saj to pomeni samo olajšavo pri računalnikih. Pri telefonih in tabličnih računalnikih navadno uporabljamo kar virtualno tipkovnico. Lahko pa uporabnikom olajšamo vnašanje teksta in povsod, kjer se le da, dodamo predvidevanje vnosa in ponudimo zadnje, oziroma pogosto uporabljene vnose, da jih uporabniki samo izberejo s seznama in jih tako natipkajo samo enkrat (slika 2.14).



Slika 2.14: Predvidevanje vnosa, ki uporabniku omogoča čim manj tipkanja.

To je bolj ali manj vse v zvezi s komunikacijo v smeri od človeka k napravi, a potrebujemo tudi nekaj komunikacije v nasprotni smeri. Uporabnik namreč želi imeti povratno informacijo na svoj dotik, da ve, da je naprava dotik pravilno zaznala. Elementi, ki jih izberemo, tako lahko spreminjajo barvo, velikost, položaj, prosojnost in podobno. Gumbi se naprimer spreminjajo glede na stanje in ob premikanju objektov po zaslonu le ti sledijo prstu, kot je prikazano na sliki 2.15. Tako lahko uporabnik hitreje in bolj zanesljivo

uporablja aplikacijo.



Slika 2.15: Povratna informacija uporabniku s spreminjanjem barve, velikosti ali premikom.

2.4 Prilagajanje platformi

Na kratko lahko omenimo še težavo prilagajanja platformi. Kot že rečeno, obstajajo razlike med platformami, zaradi katerih bi bilo bolje prilagoditi aplikacijo za vsako posebej, vendar to v poslovnem svetu ni vedno rešitev, če želimo čim nižje stroške razvoja. Sicer bo o prilagajanju platformi več govora

v naslednjem poglavju, saj je to bistvo dileme ali se odločimo za aplikacijo prilagojeno posamezni platformi (angl. native) ali za spletno aplikacijo, a vseeno želim poudariti nekaj razlik v zgradbi aplikacije.

Pravila zgradbe aplikacije se med platformami ne razlikujejo toliko, se pa razlikujejo v nekaterih kretnjah, kot smo že omenili in različnih detajlih, kot je pozicija osnovnega menija, ki se v Android aplikacijah naprimer nahaja na vrhu, pri iOS aplikacijah pa na dnu, saj imajo telefoni z operacijskim sistemom Android ponavadi na dnu vrstico s fizičnimi gumbi. Apple je med drugim veliko bolj natančen glede samega izgleda aplikacije, sicer pa obstaja še veliko podobnih podrobnih pravil glede sestave aplikacij, ki se jih je dobro držati zato, da izboljšamo uporabniško izkušnjo in naredimo aplikacijo bolj domačo [2].

Poglavje 3

Tehnološke značilnosti aplikacij za zaslon na dotik in orodja za razvoj

3.1 Uvod

Ko se odločimo, da bomo naredili mobilno aplikacijo, se pojavi vprašanje, katero tehnologijo za to uporabiti. Seveda je to odvisno od vsakega primera aplikacije posebej in eno prvih vprašanj, ki si jih pri tem moramo postaviti, je, na katerih napravah želimo aplikacijo uporabljati. Seveda pa se poleg tega pojavlja tudi vrsta drugih razlogov, zaradi katerih se odločimo za eno ali drugo tehnologijo. Med drugim so torej pomembna vprašanja cene razvoja, tehnologij, ki jih že poznamo in znamo uporabljati, nenazadnje pa seveda tudi tematika same aplikacije, saj kot bomo videli v nadaljevanju, tudi ta vpliva na nekatere odločitve pri izbiri tehnologije.

Slika 3.1: Dilema med aplikacijo, namenjeno specifično za posamezen operacijski sistem, in spletno aplikacijo. Vir: [5]

3.2 Platformi prilagojena aplikacija ali spletna aplikacija?

Ena od vročih tem na spletu v zadnjih letih je dilema platformi prilagojenih aplikacij (angl. native) proti spletnim aplikacijam. Zdi se, da ima na spletu vsak bloger in vsaka računalniško usmerjena revija svoje mnenje no tej temi in seveda se argumenti in zaključki zelo razlikujejo. Po prepričanjih podjetja GIA (Global Intelligence Alliance) naj bi naprimer v prihodnje vse aplikacije prešle v spletno verzijo (slika 3.2, [4]), pri reviji The Guardian zaenkrat verjamejo v hibridni način [12], nekateri pa verjamejo, da platformi prilagojene aplikacije še vedno ponujajo boljšo možnost [19].

Pri vsem skupaj gre v osnovi za to, ali bomo pisali za vsako platformo svojo prilagojeno aplikacijo, ali pa bomo napisali eno spletno aplikacijo, ki se bo lahko izvajala na vseh platformah. Za platformi prilagojene aplikacije

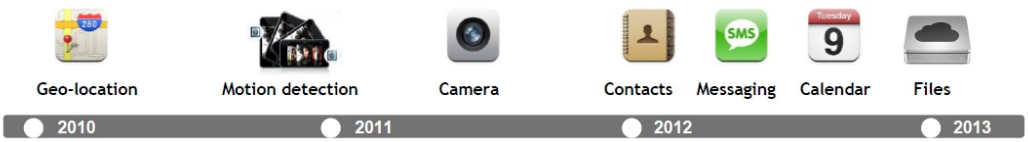
Slika 3.2: Aplikacije na vseh napravah naj bi v prihodnosti po nekaterih napovedih postale spletne aplikacije Vir: [4]

je med drugim značilno, da jih moramo pred uporabo naložiti na svojo napravo, tipično prek posebne trgovine, kot je naprimer App Store za Apple. Taka aplikacija se nato izvaja na sami napravi in večinoma ne potrebuje povezave s spletom, razen za osveževanje vsebine, posodobitve ali komunikacijo z drugimi uporabniki te aplikacije [1].

Za spletne aplikacije pa je značilno, da se večina funkcionalnosti vsakič ponovno naloži prek spleta, kar je glavna značilnost, ki loči platformi prilagojene aplikacije od spletnih aplikacij. Sicer pa so spletne aplikacije v resnici spletne strani, lastnosti, ki določajo, da je neka spletna stran v resnici spletna aplikacija, pa zaenkrat še niso dobro določene. V osnovi velja, da je spletna aplikacija tista spletna stran, ki uporabniku ponuja funkcije, ki so večinoma na voljo v obliki naloženih aplikacij. Primer je recimo spletna aplikacija Google Docs, ki nam ponuja podobne funkcije kot naprimer Microsoft Office Word, ki je naložen na operacijski sistem. Poleg tega je lahko za uporabnika spletna aplikacija na pogled skoraj enaka kot platformi prilagojena aplikacija, saj si lahko na svoj domači zaslon pripne zaznamek, ki ga ob kliku popelje na stran spletne aplikacije, kar je enako kot zagon platformi prilagojene aplikacije. Druge lastnosti, ki določajo spletno aplikacijo, pa so, da je čim bolj izolirana oziroma samozadostna, da ima bogat uporabniški vmesnik, da je

bolj kot informativno usmerjena v različna orodja in akcije in da lahko deluje tudi v načinu brez povezave s spletom [8].

Pred prihodom Html5 je bila razlika med spletnimi in platformi prilagojenimi aplikacijami za uporabnika lahko zelo očitna, saj spletne aplikacije niso imele dostopa do različnih atributov naprav, kot so naprimer kamera, pospeškomer, GPS lokator in podobni. Zato so lahko platformi prilagojene aplikacije uporabniku ponujale veliko več, vendar se to sedaj vztrajno spreminja. Slika 3.3 približno kaže dodajanje razširitev za dostop do atributov naprave. Poleg tega nam različne tehnologije (npr. Media Queries, JavaScript) omogočajo, da naredimo uporabniški vmesnik čim bolj odziven in prilagojen posamezni napravi in platformi tudi pri spletnih aplikacijah. Html5 nam omogoča tudi shranjevanje nekaterih podatkov v načinu brez povezave s spletom. Vse to pomeni, da bo za uporabnika razlika med spletno in platformi prilagojeno aplikacijo praktično nezaznavna [1].



Slika 3.3: Napoved, kdaj naj bi bile dodane razširitve za dostop do atributov naprav iz spletnih aplikacij. Vir: [4]

Ena od očitnih prednosti platformi prilagojenih aplikacij je, da imajo lahko boljšo grafično podporo in več procesorske moči, ker tečejo neposredno na operacijskem sistemu naprave. Poleg tega so lahko bolj varne, če ne potrebujejo povezave s spletom. A ravno ta povezavnost s spletom je tudi njihova slabost. Uporabniki namreč platformi prilagojene aplikacije zelo redko posodabljajo. Tudi če razvijalec popravi neko napako v aplikaciji in da na trg novo verzijo, marsikateri uporabnik aplikacije ne posodobi in ima tako slabšo izkušnjo z aplikacijo in jo slabše oceni. Pri spletni aplikaciji te težave ni, saj imajo razvijalci popoln nadzor nad verzijami in distribucijo, saj popravljanje aplikacije pomeni samo spremembo same spletne strani. Ob tem

opazimo še eno razliko. Platformi prilagojene aplikacije uporabniki nalagajo in posodabljajo prek različnih trgovin (npr. App Store, Google Play). To pomeni, da je med uporabnikom in razvijalcem nek posrednik. Ta posrednik za posredovanje aplikacije razvijalcu zaračuna neko vsoto, lahko pa ima tudi zelo ostre zahteve za aplikacijo in jo lahko tudi zavrne. Včasih preverjanje teh zahtev lahko traja tudi dalj časa, celo več kot en mesec, predvsem pri App Storeu, kjer so glede kvalitete aplikacij zelo zahtevni [1], [4]. Prednost takega posrednika pa je v tem, da taki aplikaciji zagotovi opaznost in je tako uporabnikom lažje dosegljiva. Poleg tega so aplikacije razdeljene v različne kategorije, zaradi česar primerno aplikacijo uporabniki lažje najdejo [1].

Glavna pomankljivost platformi prilagojenih aplikacij je v tem, da je potrebno za vsako platformo napisati svojo aplikacijo. Poleg tega raziskave kažejo, da podjetja večinoma nimajo dovolj znanja in zato več platformi prilagojenih aplikacij naročijo pri drugih podjetjih, spletne aplikacije pa znajo ponavadi narediti sami. Jasno je, da je zaradi tega razvoj posameznih platformi prilagojenih aplikacij seveda dražji in tudi veliko bolj zamuden. Alternativna možnost so razne hibridne tehnologije, kot je naprimer PhoneGap, ki nam omogočajo, da aplikacijo napišemo enkrat in jo nato naložimo na različne platforme. Spletne aplikacije so zato v tem pogledu boljše, saj nam omogočajo hitrejši razvoj, pri katerem napišemo samo eno verzijo aplikacije in jo samo s podrobnostmi prilagodimo posamezni platformi. Zaradi tega jih je tudi lažje vzdrževati, saj imamo vso kodo na enem mestu [4].

Nekatera podjetja to dilemo rešijo tako, da enostavno ponudijo obe možnosti, vendar je jasno, da je to najdražja možnost in si jo zato privoščijo le večja podjetja. Sicer pa trenutno prevladujejo platformi prilagojene aplikacije, vendar je po nekaterih mnenjih vzrok za to tudi boljši marketing platformi prilagojenih aplikacij in slabo tehnološko poznavanje možnosti spletnih aplikacij [1]. Če je naša aplikacija zelo zahtevna glede uporabe procesorske moči in če si želimo zelo bogat uporabniški vmesnik, potem se je bolje odločiti za platformi prilagojeno aplikacijo. Zato tudi v razredu platformi prilagojenih aplikacij prevladujejo igre, pogoste pa so aplikacije socialnih omrežij, teh-

nološko usmerjene aplikacije in aplikacije za potovanja. V razredu spletnih aplikacij pa prevladujejo vremenske aplikacije ter aplikacije različnih revij, saj vedno potrebujejo dostop do spleta [4].

3.3 Razvoj platformi prilagojenih aplikacij

Zelo očitno je v poslovnem smislu pomembno, ali se odločimo za platformi prilagojeno ali za spletno aplikacijo, saj želimo minimizirati stroške razvoja, a ponuditi čim boljši produkt uporabnikom ali naročnikom. Če se odločimo za tehnologije, specifične za posamezno platformo, je v večini primerov zelo verjetno, da bomo napisali aplikacije za vse pomembnejše platforme, razen v ekstremnih primerih, kjer aplikacijo namenimo samo določeni skupini uporabnikov, vendar je ta scenarij zelo redek, zato lahko opišemo vsaj vse glavne tehnologije, ki jih moramo v ta namen poznati. Prednosti in slabosti teh posameznih tehnologij pa nam lahko pomagajo tudi pri odločitvi med platformi prilagojeno ali spletno aplikacijo, pa tudi o tem ali bi za razvoj uporabili katerega od drugih opisanih ogrodij. V nadaljevanju so opisane splošne lastnosti razvoja aplikacij za iOS, Android in Windows 8, ki so trenutno najpomembnejše platforme.

3.3.1 Android

Aplikacije za operacijski sistem Android so napisane v Javi, kar je velika prednost, če imamo v podjetju izkušene Java razvijalce, saj to pomeni da so na domačem terenu in se razvoj Android aplikacij lahko začne veliko hitreje. Glavna razlika med operacijskim sistemom Android in operacijskim sistemom iOS je predvsem v ceni naprav na katerih sta naložena, hkrati pa tudi v njihovi raznolikosti. Operacijski sistem Android je namreč na voljo na napravah različnih proizvajalcev, pa tudi na vseh vrstah naprav, od telefonov do tabličnih računalnikov in televizijskih vmesnikov. Operacijski sistem iOS pa se nahaja le na Appleovih napravah, kar takoj zoži razpon karakteristik. To recimo tudi pomeni, da imamo manjše razlike v zaslonih, kar je prednost

ko razmišljamo o prilagajanju izgleda aplikacije zaslonu. Kako bomo razvijali svojo aplikacijo, je odvisno tudi od tega, za katero verzijo operacijskega sistema Android smo se odločili razvijati. Za razvoj aplikacij za operacijski sistem Android potrebujemo vsaj JDK (Java Development Kit), Eclipse in Android SDK (Software Development Kit) [1]. Druge podrobnosti razvoja teh aplikacij pa lahko najdemo na spletu [13].

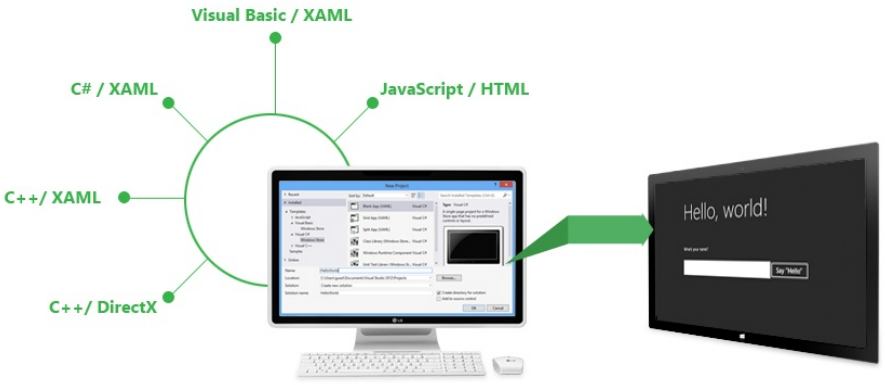
3.3.2 iOS

Nekateri menijo, da je ravno Apple prvi povzročil navušenje nad pametnimi telefoni, saj je prvi iPhone združeval veliko različnih funkcij s prijaznim in intuitivnim vmesnikom. Za razliko od operacijskega sistema Android imamo za iOS na voljo manj naprav, vendar tudi te niso vse združljive z vsemi verzijami operacijskih sistemov, zato moramo tudi tu paziti, da ne razvijamo aplikacije za iOS5, če ciljna uporabniška skupina npr. še vedno uporablja iPhone 3G. Sicer pa je začetek razvoja za iOS malce težji kot za Android. Razlog pa ni toliko v programskem jeziku, kot je v birokraciji in drugih podrobnostih, ki jih moramo urediti preden se lotimo razvoja. Najprej se začne že z izbiro opreme. Za razvoj iOS aplikacij namreč potrebujemo računalnik Mac, ki pa je večinoma precej dražji od računalnikov drugih proizvajalcev. Zatem si moramo ustvariti Appleov uporabniški račun razvijalca in kupimo iOS Developer Program [15]. Tako vidimo, da so z razvojem iOS aplikacij povezani precej višji stroški razvoja kot z razvojem aplikacij za operacijski sistem Android. Nato lahko pridobimo tudi različne certifikate, ki jih potrebujemo za podpis aplikacij, preden jih izdamo. Ko smo torej uredili vso uporabniško birokracijo, povezano z začetkom razvoja iOS aplikacij, si lahko naložimo xCode IDE in iOS SDK, ki ju potrebujemo za razvoj Objective-C aplikacij. Kot smo že omenili, pa moramo pri razvoju paziti, da je uporabniški vmesnik skladen s pravili, zahtevami in načeli, ki so opisani v posebnem dokumentu [15], saj nam aplikacijo v trgovini Apple Store sicer lahko zavrnejo. Če povzamemo, razvoj aplikacij za iOS ni ravno najbolj preprost, saj Apple na več načinov otežuje delo razvijalcem, a se lahko zanesemo, da ima Apple

zaenkrat trdno prihodnost, kar je dober argument za razvoj aplikacij za iOS [1].

3.3.3 Windows 8

Do sedaj je bil na pametnih telefonih na voljo Windows Phone 7, a novejši Windows 8 pokriva vse naprave od telefonov do prenosnikov, zato se lahko osredotočimo nanj. Aplikacijam za novi operacijski sistem pravijo Windows Store aplikacije, ki tečejo v svojem oknu, ki se privzeto raztegne čez celoten ekran. Samodejno delujejo z vsemi vnosnimi napravami, od miške in tipkovnice do zaslona na dotik. Glavna razlika med pisanjem aplikacij za Windows 8 in pisanjem za Apple ali Android je predvsem v tem, da lahko te aplikacije pišemo v večih različnih programskih jezikih, kar nam olajša začetni razvoj, saj lahko uporabimo katerega od že poznanih jezikov. Na voljo imamo JavaScript s HTML/CSS ali C# z XAML ali VisualBasic z XAML ali C++ z XAML ali C++ z DirectX, slika 3.4.



Slika 3.4: Programski jeziki, ki so na voljo za pisanje aplikacij za Windows Store. Vir: [17]

Za razvoj aplikacij imamo trenutno na voljo dve brezplačni orodji, Microsoft Visual Studio Express 2012 in Blend for Visual Studio. Za njuno uporabo potrebujemo Windows 8. Pred začetkom razvoja tudi tu potrebu-

jemo licenco, a je le ta brezplačna. Tudi Windows Store ima svoje zahteve za aplikacije, ki jih moramo upoštevati, da aplikacijo sprejmejo [18]. Zaenkrat za Windows 8 še ni napisanih veliko aplikacij, a smo lahko prepričani da bo v prihodnosti s povečanjem uporabe tega operacijskega sistema njihovo število hitro naraslo [17].

3.4 Različna ogrodja za razvoj platformi prilagojenih aplikacij

Kot je bilo že večkrat omenjeno, pa je za marsikatero podjetje lažje, če bi lahko napisali le eno aplikacijo, ter jo uporabili na vseh omenjenih platformah. Poleg tega so v mnogih podjetjih ugotovili, da bi jim precej olajšalo delo, če bi za razvoj mobilnih aplikacij lahko uporabljali že obstoječa znanja, saj so nekateri ugotovili, da traja predolgo, da se programerji naučijo dobro uporabljati programski jezik kot je naprimer Objective-C za iOS. Tak razvoj nam omogočajo različna ogrodja, s pomočjo katerih je razvoj veliko hitrejši in lažji. Produkti, ki jih dobimo s takim razvojem, se sicer ne morejo popolnoma primerjati z razvojem platformi prilagojenih aplikacij. Vseeno pa s temi ogrodji dobimo vsaj zelo dobre približke, čeprav se je pokazalo, da za večino aplikacij funkcije, ki jih ponujajo ta ogrodja, popolnoma zadostujejo. Ogrodja se razlikujejo v mnogo lastnostih, od tega katere platforme podpirajo, do programskih jezikov, ki jih pri razvoju uporabljamo in končne oblike aplikacije (platformi prilagojena, spletna ali hibridna). Vse te lastnosti lahko zelo dobro primerjamo in filtriramo s posebnim primerjalnikom na spletu [20], ki nam omogoča, da poiščemo najbolj primerno ogrodje za svoje cilje.

3.4.1 Appcelerator Titanium

Veliko ogrodij omogoča razvoj aplikacij v spletnih tehnologijah, kot so JavaScript, Html in CSS. Eden od takih je tudi Appcelerator Titanium, ki nam

omogoča, da iz ene same skupne kode generiramo platformi prilagojeno aplikacijo. Podpira razvoj za platformi iOS in Android, je brezplačen in nam omogoča uporabo večine funkcionalnosti, ki so sicer na voljo platformi prilagojenim aplikacijam. Podjetje Appcelerator sicer zagotavlja, da se aplikacije obnašajo natanko tako in so enako hitre, kot aplikacije spisane posebej v Objective-C za iOS ali v Javi za Android, a je verjetno vseeno nekaj funkcionalnih razlik, ki pa očitno niso problematične, saj Titanium uporablja več kot 350.000 razvijalcev po svetu [21]. V grobem deluje tako, da osnovno JavaScript kodo prevede v .o datoteke za iOS in v .class datoteke za Android. Le te nato ponovno prevede v binarne datoteke, ki jih potem uporabimo za distribucijo aplikacije. Tako aplikacijo lahko enako kot platformi prilagojene aplikacije naložimo na App Store ali Google Play, zaradi česar so razlike med posebej spisano platformi prilagojeno aplikacijo in s Titaniumom generirano aplikacijo za končnega uporabnika neznatne [1], [21].

3.4.2 PhoneGap

Podobno kot Appcelerator Titanium, tudi ogrodje PhoneGap temelji na uporabi spletnih tehnologij JavaScript, Html in CSS, le da je končni produkt hibridna aplikacija. To pomeni, da je enako kot platformi prilagojena aplikacija naložena na napravi in jo lahko uporabljamo tudi brez internetne povezave, hkrati pa je njen uporabniški vmesnik prikazan prek posebnega spletnega pogleda in sestavljen iz Html komponent. PhoneGap ima na voljo dostop do enakih funkcionalnosti, kot jih ponuja Html5 in še nekaj dodatnih. Značilno je, da za vsako funkcionalnost (npr. GPS) skrbi enak del JavaScript kode za vse naprave in platforme. Enako kot pri Titaniumu, se tudi tu osnovna izvorna koda prevede, da dobimo binarne datoteke, ki jih naložimo na naprave [23]. Tudi PhoneGap je zelo priljubljeno ogrodje za razvoj, saj ga uporablja več kot 400.000 razvijalcev. Poleg platform iOS in Android podpira PhoneGap razvoj tudi za BlackBerry in Symbian [1], [22].

3.5 Razvoj spletnih aplikacij

Html5, JavaScript in CSS so spletne tehnologije, ki so podlaga za razvoj spletnih aplikacij. Html (HyperText Markup Language) je označevalni jezik za izdelavo spletnih strani. Najnovejša verzija je Html5, ki prinaša nekatere novosti, predvsem pa je prva verzija, ki je močno orientirana tudi v uporabo na mobilnih telefonih in tabličnih računalnikih, saj omogoča shranjevanje nekaterih podatkov tudi v načinu brez internetne povezave, dostop do GPS funkcionalnosti naprave in podobno. CSS (Cascading Style Sheets) so stilske podloge, ki skrbijo za izgled posameznih spletnih strani, oziroma dopolnjujejo Html v oblikovnem smislu. JavaScript pa je skriptni programski jezik, s katerim lahko ustvarimo dinamične spletne strani, torej dopolnjuje Html z dodajanjem dinamike in različnih interakcij. Te tri tehnologije same po sebi omogočajo razvoj spletnih aplikacij, ki jih zaradi splošne podpore mobilnih brskalnikov podpira večina mobilnih telefonov. Pri razvoju si lahko pomagamo z različnimi ogrodji. Nekaj primerov je opisanih v nadaljevanju, sicer pa imamo pregled možnih ogrodij na voljo na spletu [20]. Katerega od njih izberemo, pa je popolnoma odvisno od nas in aplikacije, ki jo želimo zgraditi.

3.5.1 jQuery Mobile

JQuery Mobile je spletno ogrodje, optimizirano za uporabo z zasloni na dotik. Bazira na dveh JavaScript knjižnicah, jQuery in jQuery UI. Ponuja nam enoten izgled na različnih platformah in tistim, ki uporabljajo jQuery, omogoča hitro učenje in hitrejši razvoj aplikacij. JQuery je sicer knjižnica, ki nam omogoča lažje reference na elemente Html dokumenta ter lažjo obravnavo dogodkov, animacij in Ajax interakcij za hitrejši razvoj. Poleg tega je jQuery Mobile kompatibilen tudi z drugimi ogrodji za razvoj aplikacij, med drugim tudi z že opisanim ogrodjem PhoneGap, [25], [24], [1].

3.5.2 Sencha Touch

Sencha Touch je ogrodje za razvoj mobilnih aplikacij, ki v celoti temelji na spletnih standardih Html5, CSS3 in JavaScript. Omogoča razvoj aplikacij za iOS, Android, BlackBerry, Kindle Fire in druge, hkrati pa omogoča platformi prilagojen izgled aplikacij z bogatim uporabniškim vmesnikom in podporo za vse splošne kretnje in dotike. Ponovno gre za JavaScript knjižnico, ki ponuja veliko različnih grafičnih komponent, optimiziranih za interakcijo na dotik. Te komponente so naprimer različni gumbi, ki se jim izgled spreminja v skladu s platformo in napravo na kateri uporabljamo aplikacijo. Poleg gumbov imamo na voljo tudi širok spekter elementov za sestavo forme, od tekstovnega polja do izbire datuma in naslova. Ponuja nam tudi različne drsnike, ikone, menije, zavihke in druge elemente, s katerimi lahko upravljamo tudi z več prsti hkrati. Če želimo, nam Sencha SDK Tools omogoča tudi, da svojo aplikacijo prevedemo v platformi prilagojeno obliko in jo tako posredujemo tudi naprimer na App Store, [27], [1].

3.5.3 Wink

Še eno podobno ogrodje je Wink Toolkit, JavaScript knjižnica, ki nam olajša ravnanje s funkcionalnostmi naprav in premosti razlike med platformami. Ponuja vse osnovne funkcionalnosti, ki jih potrebujemo pri razvoju, od obravnave dotikov zaslona do upravljanja z objekti. Na voljo je veliko različnih komponent za sestavo uporabniškega vmesnika. Podobno kot jQuery Mobile, lahko tudi Wink uporabimo v kombinaciji z drugimi ogrodji, kot je PhoneGap. Uporabniško izkušnjo lahko izboljšamo z uporabo 3D komponent, ki so nadgradja CSS3 3D transformacij. Wink nam omogoča tudi preprost dostop do funkcionalnosti kot je GPS in pospeškomer neodvisno od naprave in platforme. Za bogatejši izgled lahko izbiramo med različnimi temami, ki jih ponuja, poleg tega pa imamo na voljo tudi predvajalnike za video in audio posnetke, ki jih lahko vključimo v svojo aplikacijo, [28].

3.6 Balsamiq Mockups - program za načrtovanje izgleda aplikacije

Preden se lotimo dejanskega razvoja aplikacije, moramo čim bolje definirati njen izgled in sestavo, zato da imamo pred seboj jasno sliko končnega produkta. Za ta namen priporočamo uporabo programa Balsamiq Mockups, s katerim lahko brez težav načrtujemo vse vrste aplikacij. Program je plačljiv in je na voljo v namizni in spletni obliki, lahko pa ga dodamo tudi kot vtičnik v Google Drive, Confluence in podobno.

Izgled aplikacije sestavimo z različnimi komponentami, ki so na voljo, ter jim po svoje prilagodimo velikost, barvo in vsebino. Program nam omogoča tudi to, da posamezni komponenti dodamo povezavo tako da ob kliku odpremo drug pogled. Izgled komponent je zelo splošen in ni namenjen za prikaz podrobnih oblikovnih lastnosti aplikacije, ampak za osnovno sestavo in položaj komponent.

Ko končamo z oblikovanjem izgleda aplikacije, se lahko s kliki premikamo med pogledi in tako dobimo zelo dober občutek končnega izdelka [29].

Poglavje 4

Primer prepisa obstoječe aplikacije v mobilno aplikacijo

4.1 Uvod - od tipkovnice in miške do zaslona na dotik

Osnovni namen tega praktičnega primera je pokazati, kako se lahko lotimo razvoja aplikacije za zaslon na dotik, če že imamo na voljo obstoječo aplikacijo, namenjeno za uporabo s tipkovnico in miško. Za primer bomo vzeli obstoječo aplikacijo, katere lastnik je podjetje Sandoz. Končni cilj tega praktičnega primera je prikazati postopek, ki nam omogoča, da večino obstoječih poslovnih aplikacij prepišemo v mobilno obliko. Nastala aplikacija je namenjena za demonstracijo in učenje postopkov razvoja podobnih aplikacij. Primer začnemo s pregledom obstoječe aplikacije in načrtovanjem izgleda ter funkcionalnosti nove aplikacije. Nato je opisano izbiranje primernega ogrodja oziroma tehnologij, s katerimi najlažje realiziramo zastavljene cilje. Sledi sam razvoj in opis poteka nastajanja nove aplikacije, ter predstavitev končnega izdelka. Za konec sledi še analiza razvoja, primerjava aplikacij in zaključni komentar.

4.2 Obstoječa aplikacija

Naša naloga je bila pretvoriti obstoječo aplikacijo v mobilno aplikacijo, oziroma v aplikacijo namenjeno za zaslon na dotik. Za ta namen smo dobili aplikacijo WebEZE podjetja Sandoz. Gre za preprosto aplikacijo namenjeno evidenci delovnega časa zaposlenih. Za obstoječo aplikacijo ni bila na voljo razlaga pomena posameznih podatkov, ki jih prikazuje aplikacija, niti razlaga delovanja. Zaradi tega je opis obstoječe aplikacije grob, oziroma pojasnjuje tiste podrobnosti, ki so pomembne za razvoj nove aplikacije, kar je dovolj za demonstracijo.

Obstoječa aplikacija je sestavljena iz treh glavnih akcij, v nemščini poimenovanih *Personalblatt*, *ZAG/Urlaubsstand* in *Zeitnachweis*. Začetno stran lahko vidimo na sliki 4.1.

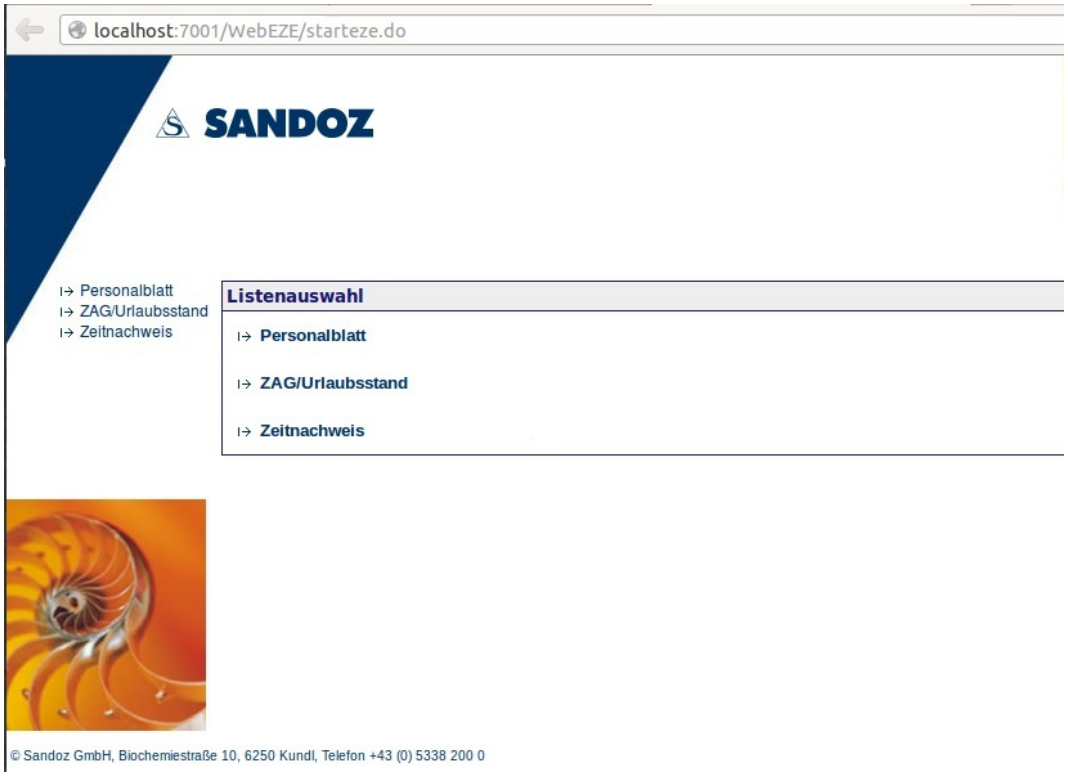
Ob izboru katerekoli od treh akcij pridemo najprej na stran za izbor zaposlenega, ki ga lahko izberemo glede na osebne podatke, osebno številko ali stroškovno oziroma delovno mesto. Izgled strani za izbor zaposlenega je na sliki 4.2.

Ob pritisku na gumb za potrditev dobimo izpis podatkov za izbranega zaposlenega. Pri akciji *Personalblatt* dobimo v pregled tri različne tabele. Prva tabela vsebuje po mesecih razporejene podatke o dnevih, kjer številka v polju predstavlja število ur, oznaka na njeni desni pa tip porabljenih ur, kjer je D oznaka za dopust, I za izobraževanje in tako naprej. Druga tabela vsebuje seštevek posameznih tipov ur po mesecih. Tako lahko vidimo, da je zaposleni januarja porabil 22 ur dopusta, razdeljenih na 8 dni. Tretja razpredelnica pa vsebuje število nadur, ki jih je zaposleni oddelal v posameznem mesecu. Nadure so dveh različnih tipov, tipa 50 in tipa 100, zadnja vrstica pa vsebuje seštevek teh nadur. Vse omenjeno je razvidno na sliki 4.3.

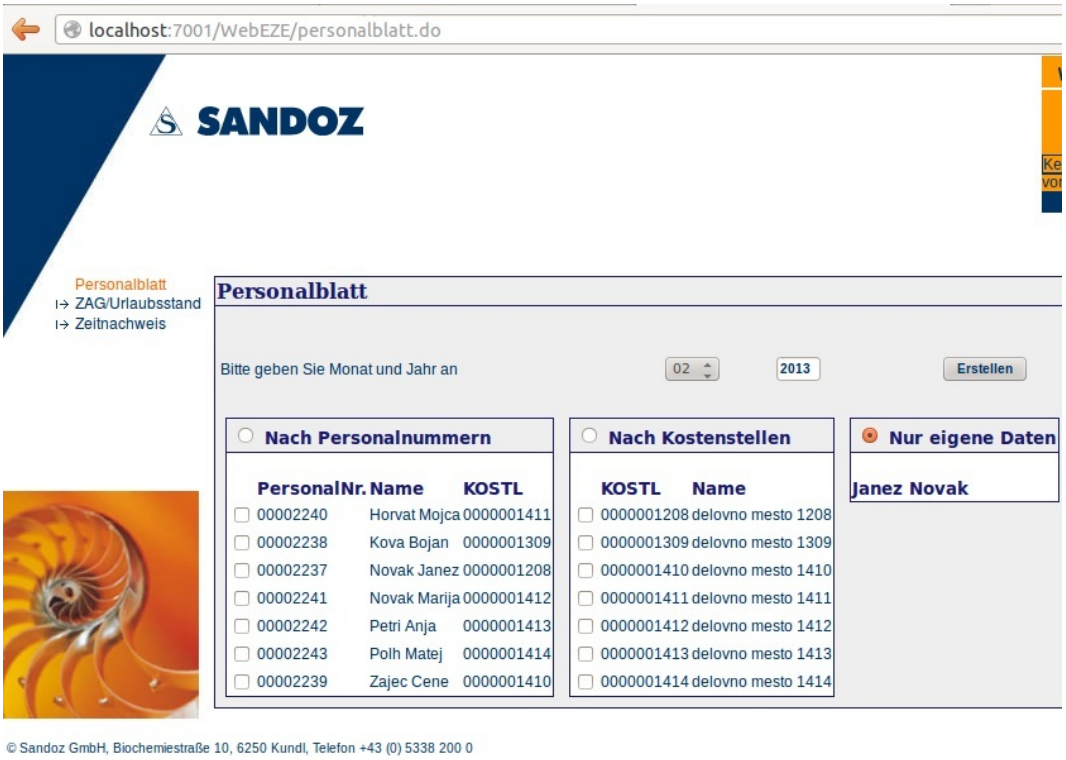
Druga akcija je *Zag/Urlaubsstand* pri kateri dobimo izpis porabljenega dopusta in nadur po dnevih za posamezni mesec. V prvi vrstici imamo začetno stanje porabljenega dopusta in nadur, pred začetkom izbranega meseca, nato po dnevih razdeljene zapise in nato še končno oziroma skupno stanje porabljenega dopusta in zbranih nadur. Opisana razpredelnica je na

sliki 4.4.

Zadnja od treh akcij je *Zeitnachweis*, ki prikaže za določen mesec točen čas prihoda, odhoda in število ur dela po posameznih dnevih. Poleg tega prikaže tudi oznako terminala, na katerem se je zabeležil čas in dnevni program. Prikaz je na sliki 4.5.



Slika 4.1: Začetna stran aplikacije WebEZE. Aplikacijo sestavljajo tri akcije: *Personalblatt*, *ZAG/Urlaubsstand* in *Zeitnachweis*.



Slika 4.2: Stran za izbiro zaposlenega. Na voljo imamo možnost izbire glede na osebno številko, delovno mesto in osebne podatke.

localhost:7001/WebEZE/personalblatt_report.do

SANDOZ

Personalblatt
↳ ZAG/Urlaubsstand
↳ Zeitnachweis

↳ PDF/drucken

Personalblatt 29.02.2012 Sandoz GmbH

Personalnummer: 00002237 **Janez Novak** **Kostenstelle: 00000012**

Url. 2012 190,00 h ANSPR. 01.03.2001 **REST 29.02.2012 550,74 h Zeitguth. per 29.02.2012**

BERECH. 01.03.1999 **REST 31.12.2011 497,54 h**

Ersatzruhe: 0,07 h

	Jänner		Februar		März		April		Mai		Juni		Juli		August		September		Oktob
01					6,00	D							6,00	D					
02			11,00	D							11,00	D							11,0
03	1,00	D							1,00	D							1,00	D	
04							10,00	D							10,00	D			
05					9,00	D							9,00	D					
06			12,00	D							12,00	D							12,0
07	3,00	D							3,00	D							3,00	D	
08							8,00	D							8,00	D			
09					4,00	D							4,00	D					
10			7,00	D							7,00	D							7,0
11	5,00	D							5,00	D							5,00	D	
12							2,00	D							2,00	D			
13					6,00	D							6,00	D					
14			11,00	D							11,00	D							11,0
15	1,00	D							1,00	D							1,00	D	
16							10,00	D							10,00	D			
17					9,00	D							9,00	D					
18			12,00	D							12,00	D							12,0
19	3,00	D							3,00	D							3,00	D	
20							8,00	D							8,00	D			
21					4,00	D							4,00	D					
22			7,00	D							7,00	D							7,0
23	5,00	D							5,00	D							5,00	D	
24							2,00	D							2,00	D			
25					6,00	D							6,00	D					
26			11,00	D							11,00	D							11,0
27	1,00	D							1,00	D							1,00	D	
28							10,00	D							10,00	D			
29					9,00	D							9,00	D					
30			12,00	D							12,00	D							12,0
31	3,00	D							3,00	D							3,00	D	

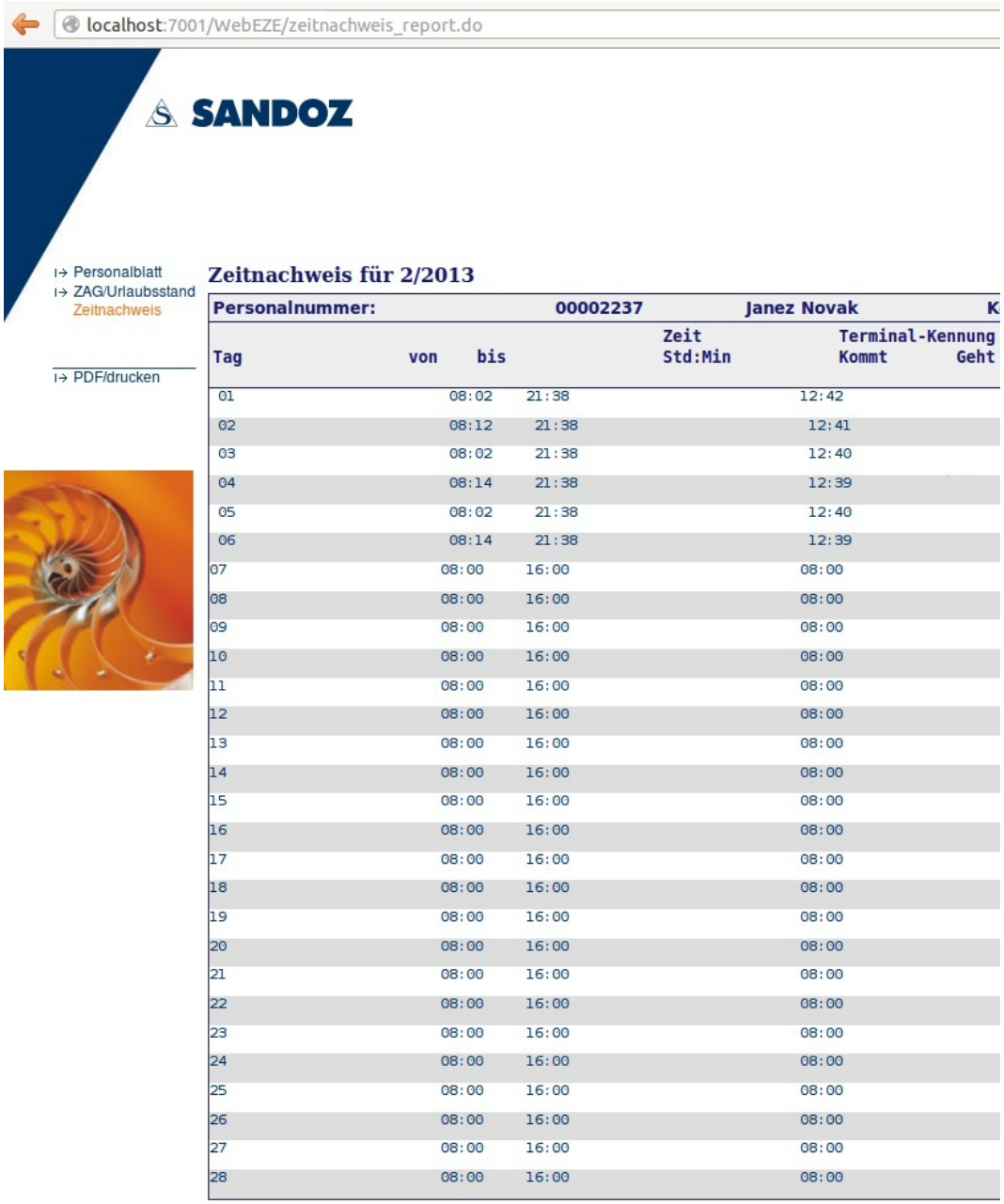
	STD	AZ	STD	AZ	STD	AZ	STD	AZ	STD	AZ	STD	AZ	STD	AZ	STD	AZ	STD	AZ	STD
B																			
D	22,00	8	83,00	8	53,00	8	50,00	7	22,00	8	83,00	8	53,00	8	50,00	7	22,00	8	83
I																			
N																			

Überstunden/Sonstige

50	10,00	2,00	8,00	4,00	3,00	12,00	9,00	5,00	7,00	
100	10,00	2,00	8,00	4,00	3,00	12,00	9,00	5,00	7,00	
SUM	20,00	4,00	16,00	8,00	6,00	24,00	18,00	10,00	14,00	

Slika 4.3: Stran za prikaz podatkov za akcijo *Personalblatt*. Prva tabela vsebuje za posamezen mesec in dan število ur posameznega tipa. Druga tabela prikazuje skupno število ur po posameznih tipih, tretja pa nadure in seštevek nadur.

Slika 4.4: Stran za prikaz podatkov za akcijo *ZAG/Urlaubsstand*. Prikazuje za izbrani mesec seznam po dnevih porabljenega dopusta in opravljenih nadur.



Slika 4.5: Stran za prikaz podatkov za akcijo *Zeitnachweis*. Prikazuje za izbrani mesec čas prihoda in odhoda, skupno število ur dela ter podatke o terminalih prihoda in odhoda.

4.3 Načrtovanje izgleda aplikacije

Kot je bilo že omenjeno, o obstoječi aplikaciji ni bilo na voljo vseh informacij, zato so nekatere podrobnosti prirejene, da bi lahko končno aplikacijo naredili čim bolj uporabno, intuitivno in razumljivo. Jasno je bilo, da dokler nimamo dokončne ideje o aplikaciji, ki jo želimo zasnovati, se ne moremo lotiti načrtovanja. Zato so v aplikaciji za zaslon na dotik izpuščene vse tiste postavke, katerih pomen ni bil jasen iz opisov, ki so bili na voljo. Nato se lahko lotimo načrtovanja izgleda z uporabo programa Balsamiq Mockups [29], pri čemer upoštevamo ugotovitve, opisane v prvem delu.

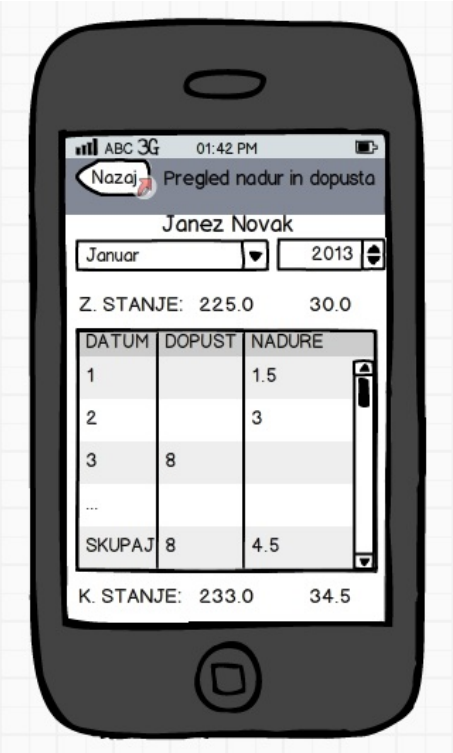
Podobno kot v obstoječi aplikaciji, tudi tu začetno stran sestavljajo trije gumbi, eden za vsako od glavnih akcij, ki pa so veliko večji in tako primerni za uporabo na zaslonih na dotik. Ob izbiri akcije, pridemo pri vseh treh enako na izbiro zaposlenega, kjer pa vidimo prvo razliko v izgledu. Pri obstoječi aplikaciji vidimo vse tri možne izbore hkrati, torej izbor po osebnih podatkih, po osebni številki in po delovnem mestu, pri aplikaciji za zaslon na dotik, pa naj bi videli le en izbor naenkrat, kar je lep primer varčevanja s prostorom na zaslonu. Med posameznimi izbori pa se lahko premikamo z izbiro pravega gumba na vrhu, slika 4.6.

Slika 4.6: Primer izgleda začetne strani aplikacije na levi z velikimi gumbi, primernimi za zaslone na dotik. Na desni je stran za izbiro zaposlenega, ki demonstrira varčevanje s prostorom z uporabo selektivnega prikaza seznamov.

Prikaz prve akcije *Mesečni pregled dela* je v obstoječi aplikaciji sestavljen iz treh različnih tabel, ki so na zaslonu prikazane hkrati. To za mobilno aplikacijo ni sprejemljivo, saj nimamo dovolj prostora na zaslonu za prikaz velikih količin podatkov. Zato smo prikaz treh tabel razdelili in je tako vsaka tabela prikazana na svojem zavihku. Poleg tega opazimo, da v obstoječi aplikaciji dobimo pregled nad celotnim letom, torej za vseh dvanajst mesecev hkrati. Tudi to ni izvedljivo na majhnih zaslonih, zato imamo pregled razdeljen na posamezne mesece in se lahko med njimi premikamo s pomočjo spustnih menijev. Vse opisano je razvidno na sliki 4.7.

Slika 4.7: Primer izgleda prikaza podatkov za akcijo *Mesečni pregled dela*. Namesto da bi vse tri tabele prikazali na zaslonu hkrati, je vsaka tabela na svojem zavihku, s čimer prihranimo prostor na zaslonu. Med posameznimi meseci se premikamo s spustnim menijem.

Prikaz druge akcije *Pregled nadur in dopustov* je pri novi aplikaciji še najbolj podoben stari, saj vsebuje le podobno tabelo, kar je razvidno na sliki 4.8.



Slika 4.8: Primer izgleda prikaza podatkov za akcijo *Nadure in dopusti*. Ta tabela je še najbolj podobna originalni, saj je dovolj ozka za prikaz na manjšem zaslonu.

Zadnja akcija *Prihodi in odhodi* pa prikaže v obstoječi aplikaciji zelo široko tabelo. Ker pri mobilni aplikaciji ne moremo zagotoviti dovolj širokega zaslona za pregledovanje celotne tabele, je smiselno, da v tabeli ne prikažemo vseh stolpcev, ampak samo najpomembnejše, do ostalih podatkov pa dostopamo z gumbom za prikaz več podrobnosti, ki odpre pojavno okno z vsemi podatki. Tudi to je odličen primer varčevanje s prostorom na manjših zaslonih. Vse to je razvidno na sliki 4.9.

Slika 4.9: Primer izgleda prikaza podatkov za akcijo *Prihodi in odhodi*. Ta tabela je sicer zelo široka, zato ne prikažemo vseh stolpcev, kar je razvidno na levi. S klikom na gumb *Več* pa odpremo pojavno okno z vsemi podatki za posamezen vnos v tabeli.

Ko izdelamo take primere izgleda aplikacije, dobimo zelo dober občutek za razvoj in zahteve mobilne aplikacije in se tako lažje odločimo med tehnologijami in drugimi orodji, ki jih bomo potrebovali za realizacijo zahtev.

4.4 Izbira tehnologije: jQuery Mobile

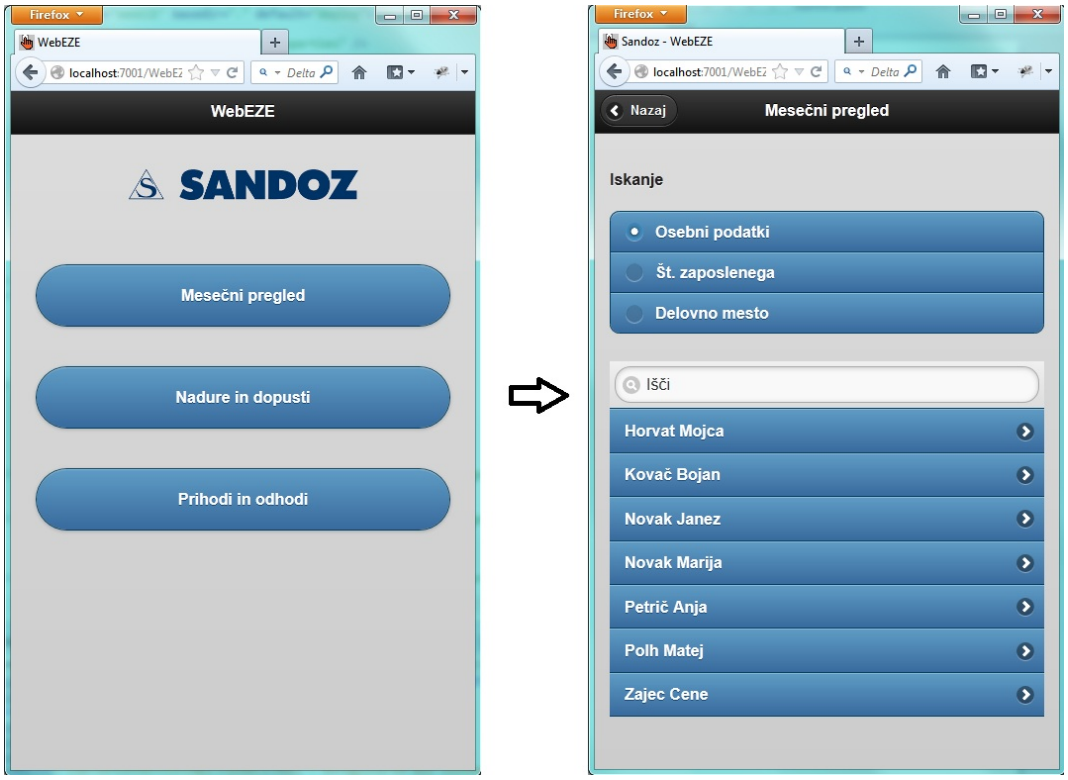
Izbira tehnologije temelji na veliko različnih dejavnikih. Nekateri od njih so trenutno poznavanje tehnologij, lastnosti aplikacije, ki jo želimo izdelati, ciljne naprave, na katerih želimo aplikacijo uporabljati, čas, ki ga imamo na voljo za razvoj in stroški, ki si jih ob razvoju lahko privoščimo. Naše zahteve pri izboru tehnologije za razvoj aplikacije WebEZE Mobile so bile, da mora aplikacija seveda teči na različnih platformah. Zaradi tega je odpadla možnost razvoja aplikacije za vsako platformo posebej, saj bi to vzelo preveč časa. Tako smo se odločili, da želimo narediti spletno aplikacijo, saj so le te najbolj fleksibilne, vse spremembe so vidne takoj, hkrati pa za njihov razvoj uporabljamo spletne tehnologije, ki jih že poznamo, kar močno zmanjša čas, ki ga potrebujemo za razvoj. Osnovna izbrana tehnologija je bil torej Html 5. Nadalje pa se je bilo potrebno odločiti še o ogrodju, ki bi ga uporabili za razvoj. V ožji izbor sta prišla jQuery Mobile in Sencha Touch, saj sta najbolj razširjena, poleg tega pa ponujata dober nabor komponent in skrbita za interakcije z zaslonom na dotik, obnašanje gumbov in preostalih komponent in nam tako prihranita skrbi z marsikatero zahtevo pri funkcionalnosti aplikacije. V nadaljevanju smo poskušali postaviti projekt z uporabo ogrodja Sencha Touch in naleteli na več težav. Če to izkušnjo primerjamo z uporabo jQuery Mobile, s pomočjo katerega smo z razvojem projekta lahko začeli v nekaj minutah, je jasno, da je jQuery Mobile tu v veliki prednosti. Poleg tega ima Sencha Touch bolj specifičen vmesnik, zaradi česar je učenje na začetku veliko počasnejše. Na podlagi teh ugotovitev smo se torej odločili, da uporabimo ogrodje jQuery Mobile. Začetek razvoja z jQuery Mobile je zelo enostaven, saj v posamezen html dokument samo dodamo reference na jQuery Mobile JavaScript datoteke in lahko začnemo z uporabo komponent. Tudi za nekoga, ki ni vešč v razvoju html strani, to ne predstavlja težav, saj lahko večino vsebine dobesedno sestavimo tako, da kopiramo html izvorno kodo z jQuery Mobile demo strani v svoj projekt in samo prilagodimo podrobnosti [26].

4.5 Razvoj in končna aplikacija

Pri razvoju aplikacije je bilo pomembno, da se v največji meri uporabi obstoječa programska koda, ter da se za spletno aplikacijo naredi le drugačen prikaz podatkov in podobno. Do podatkov se tako dostopa z istim klicem metode v obeh verzijah aplikacije. Obravnava in prikaz podatkov pa se razlikujeta. Za osnovo mobilne aplikacije je uporabljen tako imenovan jQuery Mobile Boilerplate, pripravljena osnova za vsak nov jQuery Mobile projekt [30].

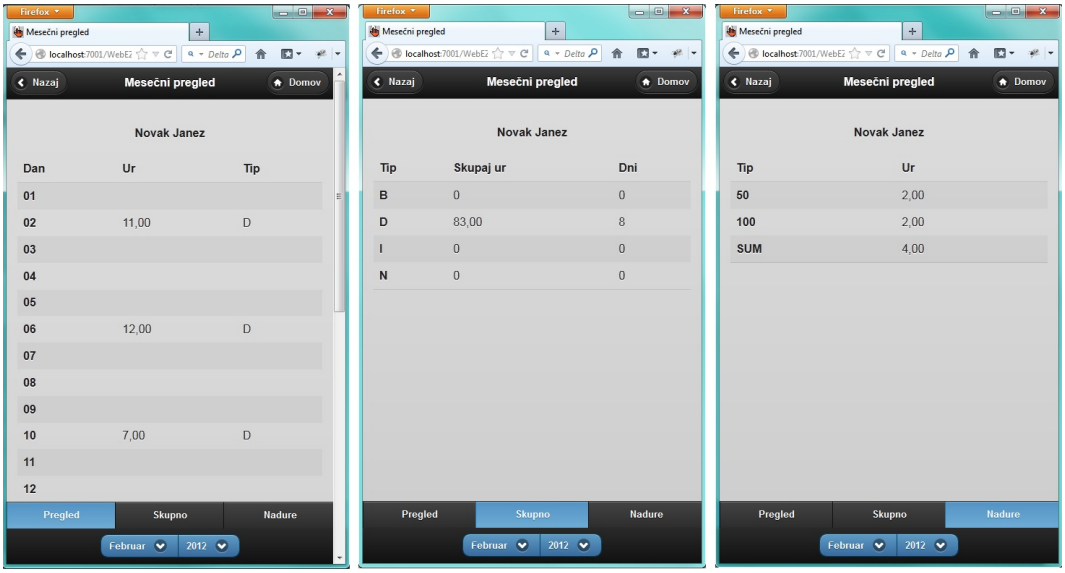
Projekt je tako sestavljen iz JSP strani, ki omogočijo dinamično generiranje spletne strani in predstavljajo osnovni prikaz aplikacije. Za obravnavanje različnih dogodkov, kot je pritisk na gumb, skrbi JavaScript, za končne podrobnosti ob prikazu strani pa se uporablja CSS. Pred podrobnejšo razlago delovanja aplikacije pa si poglejmo končni izdelek.

Na sliki 4.10 vidimo na levi začetno stran in na desni izbiro zaposlenega. Na izbiro zaposlenega pridemo s klikom na katerikoli gumb oziroma akcijo.



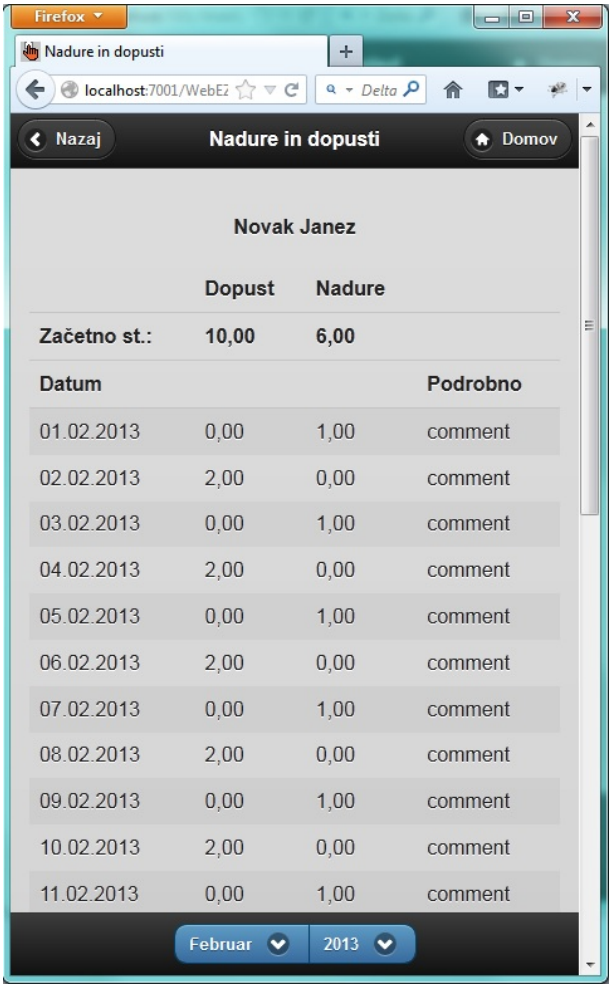
Slika 4.10: Na levi imamo začetno stran, na desni pa pogled za izbiro zaposlenega.

Pri mesečnem pregledu imamo nato tri različne zavihke. Prvi zavihek, imenovan *Pregled*, nam nudi pregled nad celotnim mesecem, s številom ur v določenem dnevu, *Tip* pa določa ali gre za dopust, izobraževanje ali podobno. Zavihek *Skupno* za izbrani mesec nudi seštevek ur za posamezni tip, zavihek *Nadure* pa skupno število različnih nadur. Vse to je vidno na sliki 4.11.



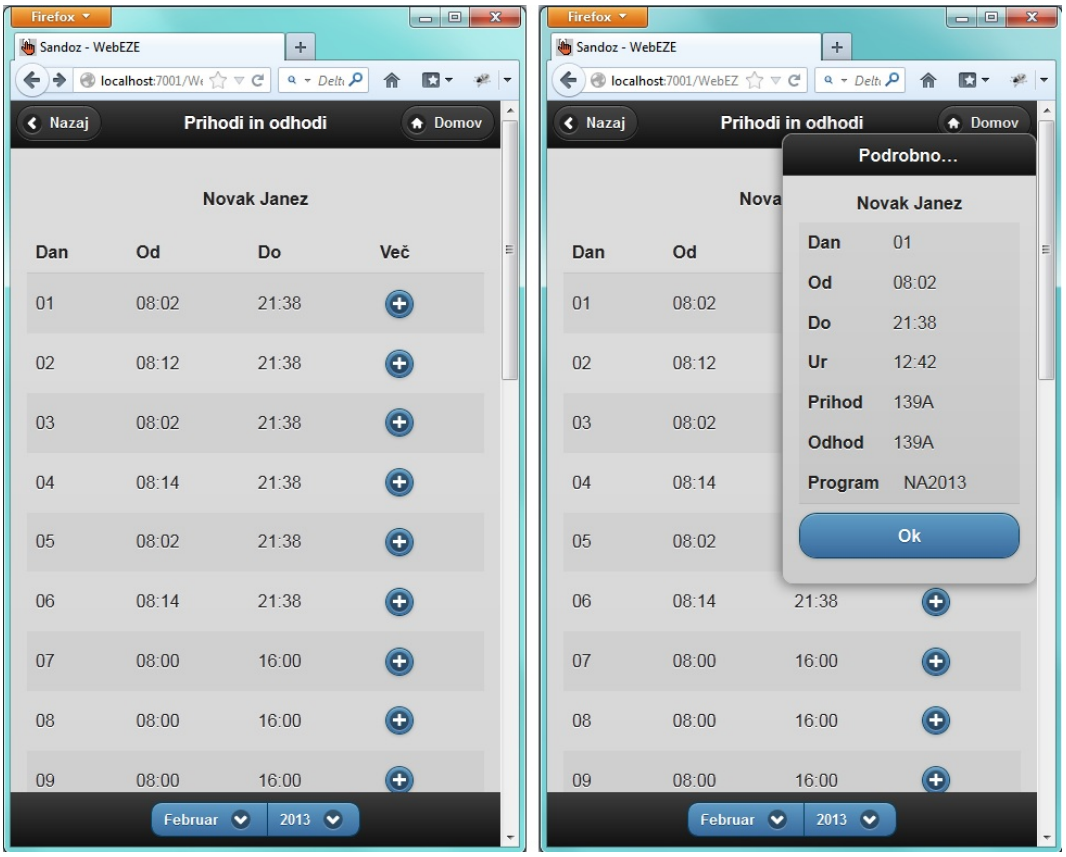
Slika 4.11: Prikaz podatkov za mesečni pregled je razdeljen na tri različne zavihke.

Druga akcija *Nadure in dopusti* nam nudi seštevek nadur in dopustov pred začetkom izbranega meseca, pregled porabljenega dopusta in opravljenih nadur po posameznih dnevih v tistem mesecu in končni seštevek. Ta pogled je viden na sliki 4.12.



Slika 4.12: Prikaz podatkov za akcijo *Nadure in dopusti*.

Zadnja akcija nam nudi pregled prihodov in odhodov po posameznih dnevih v izbranem mesecu, kjer lahko vidimo kdaj je zaposleni prišel in odšel. Ob kliku na gumb za več podrobnosti vidimo še preostale podatke, kar je vidno na sliki 4.13.



Slika 4.13: Prikaz podatkov za akcijo *Prihodi in odhodi*. Ob kliku na gumb *Več* dobimo dodatne podrobnosti za posamezen zapis v tabeli.

Sedaj si lahko pogledamo še podrobnejši primer delovanja aplikacije. Na začetni strani je gumb *Mesečni pregled* definiran tako:

```
<div class="ui-block-a">
        <a data-role="button"
            href="userSelection.do?for=personalblatt"
            data-transition="slide">
         Mesecni pregled</a>
</div>
```

Pri tem je pomemben le atribut href, ki nam pove, da se ob kliku na ta gumb izvede akcija *userSelection*. Na kratko rečeno, obstoječa aplikacija uporablja Apache Struts, zato se le ta uporablja tudi tu. V posebni konfiguracijski datoteki definiramo, kateri tako imenovani krmilnik se izvede ob tem klicu. Namen akcije je, da pridobimo podatke, s katerimi nato napolnimo stran za izbor zaposlenega. V krmilniku podatke shranimo v sejno spremenljivko in jih nato iz sejne spremenljivke preberemo na strani za izbor zaposlenega. Del kode za prikaz izbora zaposlenega je videti tako:

```
<div class="ui-grid-solo middle-container">
<div class="ui-block-a" data-content-theme="c">
        <h4>Iskanje</h4>
        <fieldset data-role="controlgroup">
                <input type="radio" name="radio-choice"
                id="radio-choice-1" value="choice-1" />
                <label for="radio-choice-1">
                Osebni podatki</label>

                <input type="radio" name="radio-choice"
                id="radio-choice-2" value="choice-2" />
                <label for="radio-choice-2">
                St. zaposlenega</label>
```

```
                <input type="radio" name="radio-choice"
                id="radio-choice-3" value="choice-3" />
                <label for="radio-choice-3">
                Delovno mesto</label>
        </fieldset>
        <br>

        <div id="l1">
        <ul data-role="listview" data-filter="true"
        data-filter-placeholder="Isci" id="list-1"
        name="list-1">
         <%
for (int i=0; i<ezeUser.getPersonnelRecords().length;
i++){
PersonnelRecord tmpPersonnelIds =
ezeUser.getPersonnelRecords()[i];
         %>
          <li>
          <a href="<%=action%>?group1=<%=i%>">
          <%=tmpPersonnelIds.getName()%>
          </a></li>
    <%
         }
    %>
        </ul>
        </div>

        <div id="l2">
        <ul data-role="listview" data-filter="true"
        data-filter-placeholder="Isci" id="list-2"
        name="list-2" >
```

```
          <%
for (int i=0; i<ezeUser.getPersonnelRecords().length;
i++){
PersonnelRecord tmpPersonnelIds =
ezeUser.getPersonnelRecords()[i];
          %>
            <li>
            <a href="<%=action%>?group2=<%=i%>">
            <%=tmpPersonnelIds.getPernr()%> -
            <%=tmpPersonnelIds.getName()%>
            </a></li>
      <%
          }
      %>
          </ul>
          </div>

          <div id="l3">
          <ul data-role="listview" data-filter="true"
          data-filter-placeholder="Isci" id="list-3"
          name="list-3" >
           <%
for (int i=0; i<ezeUser.getKostenstellen().length;
i++){
KostenstelleVO tmpCostLocations =
ezeUser.getKostenstellen()[i];
          %>
            <li>
            <a href="<%=action%>?group3=<%=i%>">
            <%=tmpCostLocations.getNummer()%>      -
            <%=tmpCostLocations.getText()%>
```

```
            </a></li>
      <%
            }
      %>
          </ul>
          </div>
</div>
</div>
```

Kot vidimo, stran sestavljajo trije gumbi za izbor sortiranja. Zaposlenega tako lahko poiščemo na podlagi osebnih podatkov, številke zaposlenega ali delovnega mesta. Kljub temu, da stran sestavljajo vsi trije seznami, pa je vedno viden le eden, odvisno torej od izbrane opcije. Za skrivanje in prikaz pravega od treh seznamov, pa poskrbi JavaScript koda, ki v primeru klika na prvi gumb izgleda tako:

```
$("#radio-choice-1").click(function() {
                    if (localStorage)
                    {
                    localStorage.setItem( 'radio', 1);
                    }
                      showlist1();
          });
function showlist1(){
          $("#l1").show();
          $("#l2").hide();
          $("#l3").hide();
          $("#radio-choice-1").attr("checked",true)
          .checkboxradio("refresh");
          $("#radio-choice-2").attr("checked",false)
          .checkboxradio("refresh");
          $("#radio-choice-3").attr("checked",false)
          .checkboxradio("refresh");
```

```
};
```

Tako se ob kliku na posamezne gumbe menjavajo vidni seznami, vsak od elementov seznama pa je gumb, ki ob kliku odpre prikaz izbranih podatkov. Vse ostale akcije in prikazi v aplikaciji so realizirani na podoben način.

Poglavje 5

Primerjava obeh rešitev

Sedaj imamo na voljo obe aplikaciji in ju lahko primerjamo. Aplikaciji WebEZE in WebEZE Mobile se v funkcionalnosti ne razlikujeta dosti. Mobilna aplikacija vsebuje vse tri akcije, enako kot osnovna aplikacija. Izbira zaposlenega in druge aktivnosti so realizirane enako in na način, ki uporabniku omogoča hitro uporabo tudi na manjših zaslonih. Razlike se pokažejo v prikazu podatkov, saj zaradi pomanjkanja podrobnejših informacij nekateri podatki v mobilni aplikaciji niso prikazani, a bi se jih dalo dodati brez posebnega truda in tudi ne bi vplivali na videz in preglednost aplikacije. Tabele in podatki, ki so prikazani v mobilni aplikaciji, sicer niso prikazani na povsem enak način kot v običajni aplikaciji, vendar so spremembe smiselne in ne vplivajo dosti na preglednost in dostopnost podatkov. Aplikacija WebEZE je sicer dober primer poslovne aplikacije, saj vsebuje veliko tabel in prikaže velike količine podatkov naenkrat. Tako vidimo, da se da tudi take aplikacije uspešno pretvoriti v mobilno obliko in prikazati podatke na manjših zaslonih. Morda to ni vedno mogoče narediti enako efektivno, ali pa zahteva več pozornosti pri načrtovanju, a je izvedljivo.

Obe aplikaciji imata svoje prednosti in slabosti, mobilna aplikacija verjetno deluje malo počasneje, odvisno od načina prenašanja podatkov oziroma hitrosti internetne povezave. Zaradi novejše izvedbe je mobilna aplikacija bolj pregledna in intuitivna, sicer pa bi se bolj pomembne prednosti in slabosti

pokazale šele pri uporabi z resničnimi podatki. Večinoma najbolj pomemben dejavnik za podjetja pri odločanju o razvoju mobilne verzije aplikacije pa so stroški, ki jih imajo pri tem. WebEZE Mobile je spletna aplikacije, ki jo podpirajo praktično vse naprave, oziroma vsaj tiste z brskalnikom in internetno povezavo. Za njen razvoj smo uporabili brezplačne tehnologije, razen programa Balsamiq Mockups za načrtovanje izgleda aplikacije, ki je plačljiv, a je njegova uporaba povsem neobvezna. Edini stroški, ki se pri takem razvoju pojavijo, so stroški, ki jih imamo z razvijalci. V tem primeru pa ključno vlogo igra čas, ki ga le ti porabijo za razvoj take aplikacije. Kot je bilo že omenjeno, lahko z razvojem aplikacije z jQuery Mobile začnemo takoj in ne izgubljamo veliko časa na začetku s postavljanjem okolja in podobnim. Tudi razvoj kasneje je hiter, tudi če imajo razvijalci samo osnovno poznavanje spletnih tehnologij. Časovno nam tako razvoj take aplikacije ne vzame veliko. Za načrtovanje, pregled in razvoj opisane aplikacije sta bila potrebna manj kot dva tedna, kar ni veliko, če upoštevamo še privajanje tehnologijam in podobno. Razvoj take aplikacije je torej hiter, kar pomeni da so tudi stroški razvoja razmeroma nizki. Za tista podjetja, ki že imajo običajno aplikacijo, je razvoj mobilne še lažji, saj lahko podobno kot jaz v tem primeru uporabijo določene dele že obstoječe programske kode za pridobivanje podatkov.

Končno vprašanje je torej, kdaj je razvoj mobilne aplikacije ekonomsko upravičen in kdaj ne. Seveda je vse to v prvi vrsti odvisno od denarja, ki ga ima neko podjetje za to na voljo, takoj nato pa je to odvisno od same aplikacije. Nekatere aplikacije so seveda lažje za pretvarjanje v mobilno obliko kot druge. Eden od dejavnikov je količina podatkov, ki se mora prenašati med spletnimi strežniki in mobilno napravo. Drug vidik je tudi ta, kako velike količine podatkov ali velike tabele razdeliti na manjše kose, ki jih lahko prikažemo na manjših zaslonih. Spet je to odvisno od vsake aplikacije posebej, pri nekaterih je to lažje storiti kot pri drugih. Pri aplikaciji WebEZE se je izkazalo, da je bilo to precej lahko storiti. Sicer pa velja, da tudi če lahko tabele smiselno razdelimo na manjše podcelote, se lahko zgodi, da je teh

podcelot preveč in imamo tako pregloboko hierarhijo pri prikazovanju, kar naredi mobilno aplikacijo manj pregledno. Nenazadnje pa velja tudi razmisliti o tem, kdo in koliko bi uporabljal mobilno aplikacijo. Nekatere aplikacije so takega tipa, da jih uporabniki večinoma potrebujejo le doma ali v službi, kjer uporabljajo aplikacijo prek računalnika, nekatere pa so izključno take, da jih uporabniki potrebujejo, ko so na poti. Od tega je tudi odvisna smiselnost pretvarjanja aplikacije v mobilno verzijo.

Zaradi vsega naštetega težko postavimo enoličen odgovor na to, ali je smiselno imeti mobilno aplikacijo ali ne. A tudi če ugotovimo, da za neko aplikacijo ni smiselno imeti mobilne verzije, pa je dobro obdržati idejo v glavi, saj se lahko v prihodnosti zgodi, da bodo nekatera podjetja uporabljala tablične računalnike na delovnem mestu, ali pa želela imeti verzijo aplikacije, prirejeno predvsem za prenosnike z zaslonom na dotik. Takrat pa ta znanja in ideje pridejo prav, tako da lahko rečemo da je razvoj mobilne verzije odvisen tudi od prihodnosti nekega podjetja in njihovega načina dela.

Poglavje 6

Zaključek

Cilj tega besedila je bil pokazati, kako lahko neko aplikacijo, namenjeno za tipkovnico in miško, prepišemo v mobilno obliko. Najprej smo spoznali osnovne značilnosti mobilnih aplikacij in tehnologije, ki jih uporabljamo za njihov razvoj, nato pa smo ta znanja uporabili pri razvoju dejanske mobilne aplikacije, ter izpostavili pomembne korake v razvoju. Končna aplikacija izpolnjuje enake zahteve kot začetna, ter dokazuje, da je mogoče velike poslovne aplikacije elegantno prepisati v mobilno obliko. Sicer je končna aplikacija namenjena predvsem za uporabo na zaslonih mobilnih telefonov, vendar je to posledica trenutnega stanja v svetu računalništva, saj zaslone na dotik največkrat najdemo na mobilnih telefonih. Šele sčasoma se bodo zasloni na dotik množično preselili tudi na računalnike z večjimi zasloni, a osnovna pravila razvoja aplikacij bodo ostala enaka. Na ključno vprašanje o tem, ali je smiselno poslovne aplikacije prepisati v mobilno obliko, lahko tako odgovorimo, da je za nekatere smiselno to storiti že sedaj, za tiste, ki pa tega trenutno ne potrebujejo, pa je potrebno, da sledijo trendom in so pripravljeni na trenutek, ko bodo aplikacije za zaslon na dotik postale standard tudi na večjih zaslonih v poslovnem svetu.

Literatura

[1] J. McWherter and S. Gowell, *Professional Mobile Application Development*, Indianapolis: John Wiley and Sons, 2012

[2] L. Wroblewski, *Mobile First*, New York: A Book Apart, 2011

[3] B. Bibeault and Y. Katz, *jQuery in Action*, Second Edition, Stamford: Manning Publications, 2010

[4] L. Luo, *Native or Web Application? (How Best to Deliver Content and Services to Your Audiences over the Mobile Phone)*, Global Intelligence Alliance, 2010

[5] mobiThinking, Mobile applications: native vs Web apps – what are the pros and cons?
http://mobithinking.com/native-or-web-app

[6] C. Villamor, D. Willis and L. Wroblewski, Touch Gesture Reference Guide, 2010
http://static.lukew.com/TouchGestureGuide.pdf

[7] L. Wroblewski, Design for Mobile: What Gestures do People Use? (23.9.2010)
http://www.lukew.com/ff/entry.asp?1197

[8] mobiThinking, What is a Web-based mobile application or Web app? Here's expert opinion from the W3C (13.8.2010)
http://mobithinking.com/blog/what-is-a-Web-app

[9] The Economist, Tablets from on high (27. 10. 2012)
http://www.economist.com/news/business/21565225-microsoft-makes-its-pitch-mobile-age-tablets-high

[10] Bloomberg, China Mobile Internet Use Overtakes Desktop Access (19. 7. 2012)
http://www.bloomberg.com/video/china-mobile-internet-use-overtakes-desktop-access-KqwumruMT1aqiFzdtKw7uw.html

[11] Pontiflex, About Half Of Mobile App Clicks Are Accidental (27.1. 2011)
http://paidcontent.org/2011/01/27/419-pontiflex-about-half-of-mobile-app-clicks-are-accidental/

[12] The Guardian, HTML5 and native apps: the hybrid approach (28.8.2012)
http://www.guardian.co.uk/info/developer-blog/2012/aug/28/html5-native-apps-hybrid-approach

[13] Android developer
http://developer.android.com/

[14] Android design
http://developer.android.com/design

[15] iOS Developer Program
https://developer.apple.com/programs/ios

[16] Windows, Touch interaction design
http://msdn.microsoft.com/en-us/library/windows/apps/hh465415.aspx

[17] Windows, Start here
http://msdn.microsoft.com/en-us/windows/apps/jj679957

[18] Windows, Windows 8 app certification requirements
http://msdn.microsoft.com/en-us/library/windows/apps/hh694083.aspx

[19] Eric Sink, Mobile Apps: HTML5 vs Native (27.8.2012)
http://www.ericsink.com/

[20] Mobile Frameworks Comparison Chart
http://www.markus-falk.com/mobile-frameworks-comparison-chart

[21] Appcelerator Titanium
http://www.appcelerator.com/platform

[22] PhoneGap
http://www.phonegap.com/

[23] PhoneGap Explained Visually
http://www.phonegap.com/2012/05/02/phonegap-explained-visually/

[24] jQuery Mobile
http://jquerymobile.com/

[25] jQuery
http://jquery.com/

[26] jQuery Mobile Demos
http://jquerymobile.com/demos/1.2.0/

[27] Sencha Touch
http://www.sencha.com/products/touch/

[28] Wink Toolkit
http://www.winktoolkit.org/

[29] Balsamiq Mockups
http://www.balsamiq.com/products/mockups

[30] jQuery Mobile Boilerplate
https://github.com/commadelimited/jQuery-Mobile-Boilerplate