

Dokumenten för Padel Buddy i kursen TDA367

Carl-Johan Björnson, Daniel Karlkvist, Fredrik Lilliecreutz,
Marcus Axelsson och Robin Repo Wecklauf

2019-10-25
1.0

Requirements and Analysis Document för Padel Buddy

1 Introduktion

Padel klassas som en racketsport och ses som en kombination av squash och tennis. Det spelas parvis på en plan som är cirka en tredjedel av en tennisplans storlek. Planen har väggar som spelarna kan använda på ett liknande sätt som squash. Padel använder stränglösa racket och likadana bollar som i tennis. Poängsättningen är även detsamma som i tennis. Eftersom padel kräver fyra personer strävar detta projekt mot att skapa en mobilapplikation som används till att hitta motståndare med en specifik skicklighetsnivå. Det ska även gå att bestämma när och var matchen ska spelas. Syftet med projektet är alltså att förenkla sökandet av motståndare i padel och förkorta processen vid bestämmelse av plats och tid.

Applikationen är även modulerad på ett sätt som möjliggör applikationen för utbyggnad inom andra sporter med liknande problem.

I Padel Buddy ska man som användare kunna:

- Skapa och redigera sin profil.
- Ha en skicklighetsnivå och profilvärdering.
- Skapa annonser med bestämd plats och tid.
- Filtrera annonser.
- Se sina spelade och uppkommende matcher.
- Ge omdömen om ens motståndare efter matchen.
- Fylla i hur matchen slutade.

Den huvudsakliga intressenten för denna applikation är samtliga projektmedlemmar, examinator, handledare och padel spelare. Samtliga människor som vill testa en ny sport, hitta andra att spela med eller utvecklas inom Padel kommer dra nytta av Padel Buddy.

1.1 Definitioner, akronymer och förkortningar

App - Applikationen.

Matchannons - Ett kort som finns i "Hem"-fliken och innehåller 4 st. spelare, datum, tid och plats för en match.

Skicklighetsnivå - Ett mått på hur duktig en spelare är.

Omdöme - Ett mått på hur pålitlig en spelare är. Till exempel om spelaren dyker upp i tid.

2 Krav

2.1 User Stories

I denna sektion kommer samtliga User Stories listas.

2.1.1 PB001, Flikar

1. Implementerad? Ja.
2. Beskrivning
 - Som en produktägare vill jag kunna se och kunna byta mellan de olika flikarna för att kunna navigera sig i appen.
3. Godkännande
 - Funktionell
 - Kan jag klicka på en ikon för att byta till motsvarande flik?
 - Kan jag byta till samtliga flikar oavsett vilken flik jag befinner mig i?
 - Om jag klickar på min nuvarande flik när jag redan befinner mig överst, händer då ingenting?
 - Kan jag trycka på motsvarande ikon nederst på appen för att komma högst upp på sin nuvarande flik?
 - Kan jag se att varje flik har en förklarande ikon (och text)?
 - Icke-funktionell
 - Tillgänglighet
 - a) Kan jag alltid se navigationsfältet?

2.1.2 PB002, Design

1. Implementerad? Ja.

2. Beskrivning

- Som en utvecklare vill jag se designen av applikationens olika delar så att implementationen blir lättare.

3. Godkännande

- Funktionell

– Finns det skisser på Padel Buddy i Adobe XD?

- Icke-funktionell

– Tillgänglighet

a) Är skissen alltid åtkomlig för samtliga i denna grupp?

2.1.3 PB003, Min profil

1. Implementerad? Ja.

2. Beskrivning

- Som en användare vill jag kunna se och redigera min profil för att hålla den uppdaterad.

3. Godkännande

- Funktionell

- Kan jag se mitt namn i min profil?
- Kan jag se en bild i min profil?
- Kan jag se en text i min profil?
- Kan jag se mitt omdöme i min profil?
- Kan jag se min skicklighetsnivå i min profil?
- Är funktionen för redigering lätt tillgänglig och tydlig?
- Kan jag gå in i redigeringsläge och ändra mitt förnamn?
- Kan jag gå in i redigeringsläge och ändra mitt efternamn?
- Kan jag gå in i redigeringsläge och ändra min biografi?
- Kan jag gå in i redigeringsläge och ändra min bild?

- Icke-funktionell

- Tillgänglighet

- a) Kan jag alltid gå in i redigeringsläge om jag befinner mig i "Profil"?
- b) Kan jag alltid se mitt namn om jag befinner mig i "Profil"?
- c) Kan jag alltid se min bild om jag befinner mig i "Profil"?
- d) Kan jag alltid se min biografi om jag befinner mig i "Profil"?

- Säkerhet

- a) Om jag skriver in specialtecken, förutom bindestreck, i namnfälten, kan jag inte spara då?
- b) Om jag skriver in svenska tecken (t.ex. åäö) i förnamnsfältet, kan jag spara då?
- c) Om jag lämnar ett, eller båda, namnfälten tomta, kan jag inte spara då?

2.1.4 PB004, Se matchannonser

1. Implementerad? Ja.

2. Beskrivning

- Som en användare vill jag se en annons som innehåller relevant information för att avgöra om jag vill gå med eller inte.

3. Godkännande

- Funktionell

- Kan jag se matchannonsen?
 - Kan jag se motståndarnas namn?
 - Kan jag se matchannonsons skicklighetsnivå?
 - Kan jag se motståndarnas omdöme?
 - Kan jag se datumen för matchen?
 - Kan jag se platsen där matchen ska spelas?
 - Kan jag gå med i samtliga matchannonser, utan dem jag redan är med i, och få spela med personerna som är med i den?

- Icke-funktionell

- Tillgänglighet
 - a) Kan jag se alltid en grön knapp vid namn "Gå med" i matchannonserna, som befinner i flödet, där jag själv inte är med i?

2.1.5 PB005, Hårdkodade spel & spelare

1. Implementerad? Ja.

2. Beskrivning

- Som en utvecklare vill jag kunna skapa hårdkodade spel med hårdkodade spelare för att kunna testa all funktionalitet.

3. Godkännande

- Funktionell

- Kan jag se hårdkodade spelare i olika hårdkodade matchannonser?
 - Kan jag bläddra mellan bland olika hårdkodade matchannonser och hårdkodade spelare?

2.1.6 PB006, “Kommande matcher”-flik

1. Implementerad? Ja.

2. Beskrivning

- Som en användare vill jag se mina kommande matcher under en egen flik inuti ”Spel” för att enkelt hitta samtliga matcher jag tillhör.

3. Godkännande

- Funktionell

- Kan jag se en egen flik under ”Spel” vid namn ”Kommande matcher”?
 - Kan jag trycka på ”Kommande matcher”-fliken och se samtliga matcher jag tillhör, och inte blivit spelade än?
 - Kan jag se en röd knapp vid namn ”Lämna match”?
 - Kan jag trycka på ”Lämna match” och inte tillhöra matchen längre?
 - Om jag trycker på ”Lämna match”, hamnar matchannonserna i flödet igen?

- Icke-funktionell

- Tillgänglighet
 - a) Kan jag alltid se en knapp vid namn ”Lämna match” i ”Kommande matcher”-fliken?
 - b) Kan jag fortfarande se all information som befann sig i matchannonserna när jag gick med i den?

2.1.7 PB007, "Tidigare matcher"-flik

1. Implementerad? Ja.

2. Beskrivning

- Som en användare vill jag se mina tidigare matcher under en egen flik inuti "Spel" för att hitta matcher jag spelat.

3. Godkännande

- Funktionell

- Kan jag se en flik vid namn "Tidigare matcher" under "Spel"?
 - Kan jag trycka på "Tidigare matcher" och se samtliga tidigare matcher som jag spelat?
 - Kan jag se en knapp vid namn "Rapportera resultat" inuti kortet?
 - Kan jag se ett resultat på den angivna matchen genom trycka på "Rapportera resultat"-knappen?

- Icke-funktionell

- Tillgänglighet

- a) Kan jag alltid se denna flik inut "Spel"-fliken?

2.1.8 PB008, Gå med i matchannonser

1. Implementerad? Ja.

2. Beskrivning

- Som en användare vill jag kunna gå med i matchannonser för att spela padel med personerna i annonsen.

3. Godkännande

- Funktionell

- Om jag trycker på ”Gå med”-knappen, hamnar matchannonsen i ”Kommande matcher”-fliken under ”Spel”?

- Om jag går med i matchannonsen, uppdateras då skicklighetsnivån baserat på min nuvarande nivå?

- Icke-funktionell

- Tillgänglighet

- a) Kan jag alltid se samtliga matcher jag gått med i under ”Tidigare matcher”?

2.1.9 PB009, Skapa matchannon

1. Implementerad? Ja.

2. Beskrivning

- Som en användare vill jag skapa en annons för att hitta någon att spela med.

3. Godkännande

- Funktionell

- Kan jag se en avskiljningslinje mellan spelarna jag bjudit in?
 - Kan jag se var man kan välja vilken padelarena som matchen ska spelas i?
 - Kan jag se var man kan välja vilket tid/datum matchen ska spelas?
 - Kan jag se var jag ska fylla i en beskrivning om denna match?
 - Kan jag välja mellan olika padelarenor i en lista?
 - Kan jag välja om matchen ska vara 60 minuter eller 90 minuter?
 - Kan jag välja mellan olika tider?
 - Kan jag trycka på ”Bjud in en spelare” och bjuda in hårdkodade spelare?
 - Kan jag endast bjuda in en hårdkodad spelare vid ett knapptryck?

- Icke-funktionell

- Tillgänglighet
 - a) Kan jag alltid se min bild under beskrivningen?
 - b) Kan jag alltid se min namn under beskrivningen?
 - c) Kan jag alltid se mitt omdöme under beskrivningen?
 - Säkerhet
 - a) Om jag trycker på ”Skapa annons”, men inte fyllt i padelarena, datum och tid, kan jag inte skapa en matchannon då?

2.1.10 PB010, Utbyggbar

1. Implementerad? Ja.

2. Beskrivning

- Som produktägare vill jag att appen ska vara utbyggbar med flera sporter för att nå ut till fler potentiella användare.

3. Godkännande

- Funktionell

- Kan jag implementera ett interface vid namn "IGame" så att inte vyerna har beroenden till klassen Game i modellen?
- Kan jag ändra klassen "Game" så att den är abstrakt, och innehåller det alla olika sporter innehållar?
- Kan jag implementera klassen "PadelGame" som ärver klassen "Game"?

2.1.11 PB011, Factory

1. Implementerad? Ja.

2. Beskrivning

- Som utvecklare vill jag kunna testa funktionalitet med hjälp av påhittade användare, som är avskilda från modellen för att se att logiken fungerar.

3. Godkännande

- Funktionell

- Kan jag se klasserna som har hand om detta i ett paket vid namn "Service"?
 - Kan jag skapa hårdkodade testannonser som är avskilda från modellen?
 - Kan jag skapa hårdkodade testanvändare som är avskilda från modellen?

- Icke-funktionell

- Tillgänglighet
 - a) Kan jag alltid hitta klasserna som har hand om detta i "Service", när det inte finns en databas?

2.1.12 PB0012, Välj användare

1. Implementerad? Ja.

2. Beskrivning

- Som produktägare vill jag kunna välja mellan olika hårdkodade användare när man startar applikationen

3. Godkännande

- Funktionell

- Kan jag se olika knappar som väljer vilken användare appen har för tillfället?
 - Kan jag klicka på ”Daniel”-knappen för att välja Daniel som användare?
 - Kan jag klicka på ”Robin”-knappen för att välja Robin som användare?
 - Kan jag klicka på ”Marcus”-knappen för att välja Marcus som användare?

- Icke-funktionell

- Tillgänglighet
 - a) Kan jag alltid se dessa knappar vid start av applikationen?

2.1.13 PB013, "Förfrågningar"-flik

1. Implementerad? Nej.

2. Beskrivning

- Som en användare vill jag kunna se en "Förfrågningar"-flik för att hitta samtliga förfrågningar man fått om att få gå med i ens matchnons.

3. Godkännande

- Funktionell

– Kan jag se fliken under "Spel" till vänster om "Kommande matcher"?

- Icke-funktionell

– Tillgänglighet

a) Kan jag alltid se denna flik när man är inne i "Spel"-fliken?

2.1.14 PB014, Förfrågningar

1. Implementerad? Nej.

2. Beskrivning

- Som en användare vill jag kunna se samtliga spelare som ansökt om att få gå med i min matchannonser och kunna godkänna eller avböja dessa för att välja vilka jag ska spela med.

3. Godkännande

- Funktionell

- Kan jag se en lista med spelare som ansökt att gå med?
 - Kan jag se vilken match spelaren har ansökt om att få gå med?
 - Kan jag se en röd knapp för att avböja spelaren att få gå med i matchannonserna?
 - Kan jag se en grön knapp för att godkänna spelare att få gå med i matchannonserna?
 - Kan jag hitta denna lista med spelare i fliken "Spel" under "Förfrågningar"?
 - Kan jag se spelarens namn?
 - Kan jag se spelarens bild?
 - Kan jag se spelarens skicklighetsnivå?
 - Om jag trycker på spelarens bild, kommer jag till deras profil?
 - Om jag trycker på spelarens namn, kommer jag till deras profil?

- Icke-funktionell

- Tillgänglighet
 - a) Kan jag inte längre se förfrågningen mer när jag avböjt spelaren från att gå med?

2.1.15 PB015, "Turneringar"-flik

1. Implementerad? Nej.

2. Beskrivning

- Som en användare vill jag kunna se en "Turneringar"-flik för att snabbt hitta till turneringar.

3. Godkännande

- Funktionell

– Kan jag se denna flik bredvid "Förfrågningar"-fliken?

- Icke-funktionell

– Tillgänglighet

a) Kan jag alltid se denna flik inuti "Spel"-fliken?

2.1.16 PB016, Turnering

1. Implementerad? Nej.

2. Beskrivning

- Som en användare vill jag kunna ha möjligheten att gå med i en turnering för att det är roligt.

3. Godkännande

- Funktionell

- Kan jag gå in i "Turneringar"-fliken och se en turneringsstruktur?
 - Kan jag se samtliga spelare i turneringsstrukturen?
 - Kan jag klicka på samtliga spelare i turneringsstrukturen för att komma till deras profil?
 - Kan jag se min uppkommande match både i "Turneringar"-fliken och "Kommande matcher"-fliken?
 - Kan jag se turneringens skicklighetsnivå?
 - Om jag inte har lägsta skicklighetsnivån som krävs för turneringen, kommer ingen valbar knapp upp för mig att gå med då?

- Icke-funktionell

- Tillgänglighet

- a) Kan jag alltid gå med i en turnering ifall den inte är full och jag har lägsta skicklighetsnivån som krävs?

2.1.17 PB017, Filter

1. Implementerad? Nej.

2. Beskrivning

- Som en användare vill jag kunna filtrera mellan annonser för att hitta en som passar mig.

3. Godkännande

- Funktionell

- Kan jag se en filtreringsknapp?
- Kan jag se en lista av platser i Göteborg att filtrera utefter?
- Kan jag se en lista av skicklighetsnivåer att filtrera utefter?
- Kan jag se två knappar av matchlängd att filtrera utefter?
- Kan jag se datum att filtrera utefter?
- Kan jag se ett tidsintervall att filtrera utefter?
- Kan jag filtrera matchannonserna utefter flera alternativ samtidigt?
- Om jag trycker på filtreringsknappen, kommer ett nytt kort upp med olika filtreringsalternativ?
- Om jag inte valt att filtrera matchannonserna, prioriteras (hamnar överst) dem som är närmst i tid?
- Om jag väljer att filtrera matchannonserna utefter en plats i Göteborg, filtreras dessa rätt då?
- Om jag väljer att filtrera matchannonserna utefter en skicklighetsnivå, filtreras dessa rätt då?
- Om jag väljer att filtrera matchannonserna utefter en viss längd på matchen, filtreras dessa rätt då?
- Om jag väljer att filtrera matchannonserna utefter ett datum, filtreras dessa rätt då?
- Om jag väljer att filtrera matchannonserna utefter ett tidsintervall, filtreras dessa rätt då?

- Icke-funktionell

- Tillgänglighet

- a) Kan jag alltid se en filtreringsknapp i "Hem"-fliken?

2.1.18 PB018, Kommentarer

1. Implementerad? Nej.

2. Beskrivning

- Som en användare vill jag kunna se kommentarer på min och andras profiler för att se deras omdömen från andra spelare.

3. Godkännande

- Funktionell

- Kan jag se namnet på den som kommenterade?
 - Kan jag se bilden på den som kommenterade?
 - Kan jag se när kommentaren skickades?
 - Kan jag se vad som kommenterades?
 - Kan jag se en tydlig avskiljning mellan varje kommentar?
 - Om jag trycker på namnet på den person som kommenterade, kommer jag till deras profil?

- Icke-funktionell

- Tillgänglighet
 - a) Kan jag alltid se kommentarer i en profil, om det finns några?

2.1.19 PB019, "Vänner"-flik

1. Implementerad? Nej.

2. Beskrivning

- Som användare vill jag kunna lägga till vänner för att enklare skapa annonser med dessa.

3. Godkännande

- Funktionell

- Kan jag lägga till vänner genom att söka efter namn?
 - Kan jag lägga till en vän ifrån en annons med spelare jag inte är vän med?
 - Kan jag trycka på ens nya väns namn och komma till deras profil?
 - Kan jag trycka på ens nya väns bild och komma till deras profil?

- Icke-funktionell

- Tillgänglighet
 - a) Kan jag alltid se mina vänner under denna flik?

2.1.20 PB020, Chatt

1. Implementerad? Nej.

2. Beskrivning

- Som användare vill jag kunna skriva med spelare för att lättare komma fram till informationen som ska vara med i annonsen.

3. Godkännande

- Funktionell

- Kan jag starta en chatt genom att klicka på en vän i min lista med vänner?
 - Kan jag skilja mellan mina meddelanden och vänrens meddelanden med en viss färg?
 - Kan jag skilja mellan mina meddelanden och vänrens meddelanden med var på skärmen meddelandet är positionerad i chatten?
 - Kan jag inte ändra på ett meddelande som redan är skickat?

- Icke-funktionell

- Tillgänglighet
 - a) Kan jag alltid se chattens historik?

2.2 Definition of Done

- Uppnått Acceptance criteria.
- Passerade unittester.
- Modellvis följer principen Open/Closed.
- Fått koden granskad.
- Uppdaterat JavaDoc.
- Merge med master-branch.
- Uppdaterat UML.

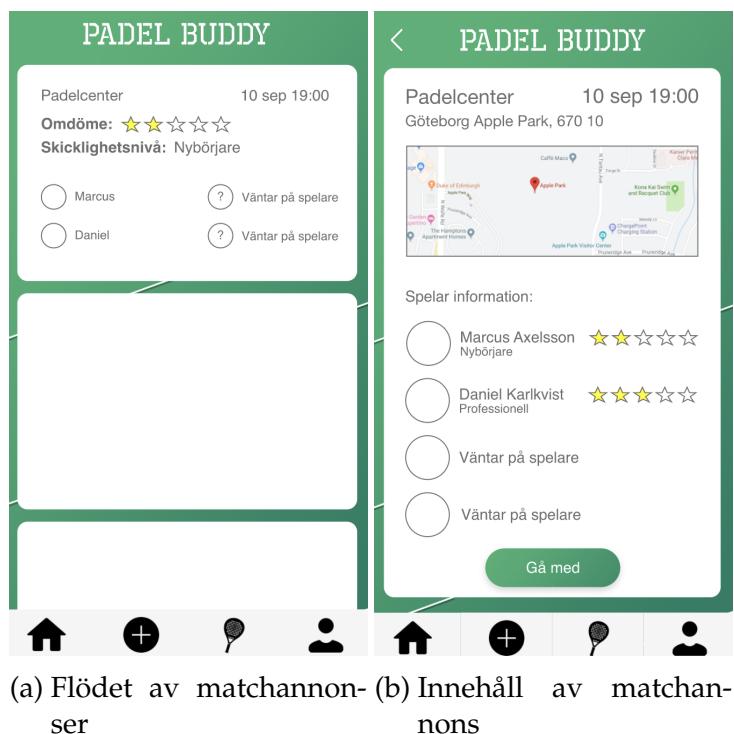
2.3 Användargränssnitt

2.3.1 Skisser

Vid start av applikationen är det tänkt att användaren hamnar på startsidan som innehåller ett flöde av matchannonser, se Figur 1a. Tanken är att användaren här kan bläddra bland olika matchannonser skapade av olika användare, inklusive sig själv om man gjort någon. Omdömet, vilka spelare som är med, hur många man väntar på, vilken skicklighetsnivå samt både var och när denna match är tänkt att spela går att se på ett tydligt sätt. Användaren kan anlända till denna vy genom att klicka på den nedersta knappen till vänster.

Vidare från Figur 1a är det tänkt att användaren kan anlända till Figur 1b genom att trycka på en matchannonser. Här visas ytterligare information om matchen som användaren inte ser vid flödet av matchannonser.

Färgschemat av grönt och vitt valdes för att det signalerar en frisk och sund känsla vid användning av appen. Vidare ville utvecklarna att användarna skulle få en sportig känsla, och känna att denna app anses vara förknippad med sport, vilket detta färgschema åstadkommer.

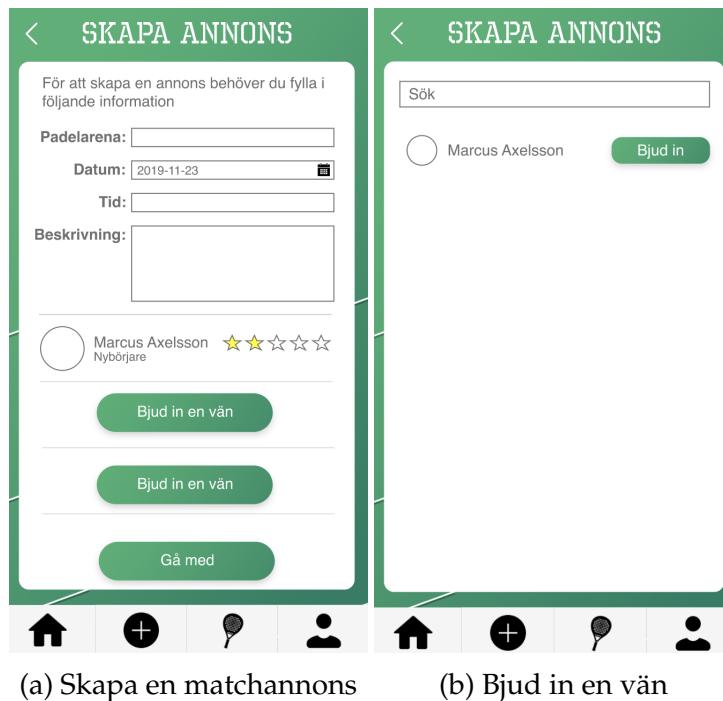


Det är tanken att användaren kan anlända till en profil-vy, se Figur 2a, genom att klicka på den nedersta knappen till höger. Där kan användaren se information som den har fyllt i, samt kommentarer som den fått av andra användare. Namnet, biografin samt bilden går även att ändra genom att klicka på redigera-knappen överst i högra hörnet.



(a) Användarprofil

Användaren ska även kunna anlända till en vy där matchannonser skapas, se Figur 3a, genom att trycka på den andra ikonen nederst till vänster. Där kan användaren sedan välja specifikationer kring matchen. Användargränssnittet är format så att användaren fyller i all information uppfirån-ner. Detta implementeras för att göra det så intuitivt som möjligt för användaren. När användaren klickar på "Bjud in en vän" så hamnar användaren till en annan vy, se Figur 9b. Där ser användaren samtliga senast spelade spelare, både med och mot. Detta gör det enkelt för användaren att, både snabbt och effektivt, bjuda in sina vänner och andra spelare.



En vy för skapade annonser och tidigare spelade matcher ska finnas tillgänglig för användaren, se Figur 4a. Där kan användaren exempelvis dubbelkolla på sitt skapade kort så all information stämmer, samt se hur man presterat i tidigare matcher.

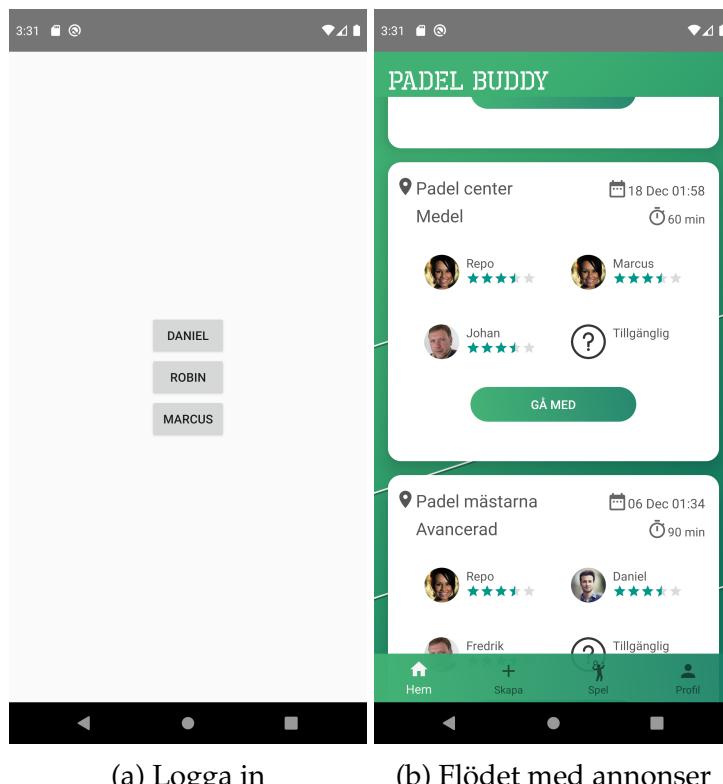


(a) Flik för matcher

2.3.2 Implementerad design

Den slutgiltiga applikationen innehåller tre hårdkodade användare som man kan logga in via, se Figur 5a. Det implementerades då man inte kan skapa några egna konton ännu.

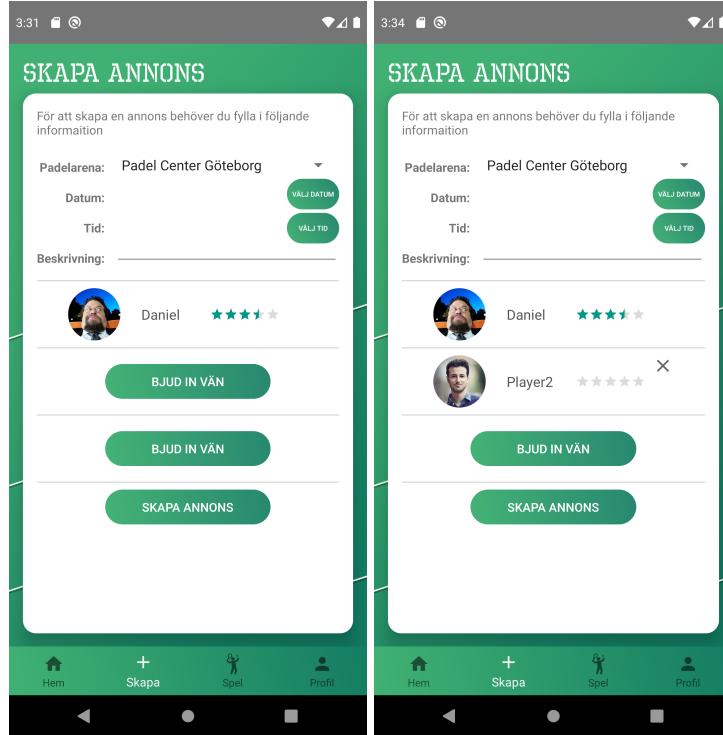
Den slutgiltiga designen av startsidan har inte ändrats jättemycket ifrån skissen, se Figur 5b. Däremot implementerades inte vyn med mer information om en annons, detta beslut togs eftersom den inte gav någon mer information än vad matchannonsen gör i flödet. Detta medförde då att "Gå med" knappen flyttades till själva matchannonsen. Omdömet är inte ett genomsnittligt värde som det var i skissen, däremot är skicklighetsnivån fortfarande det.



(a) Logga in

(b) Flödet med annonser

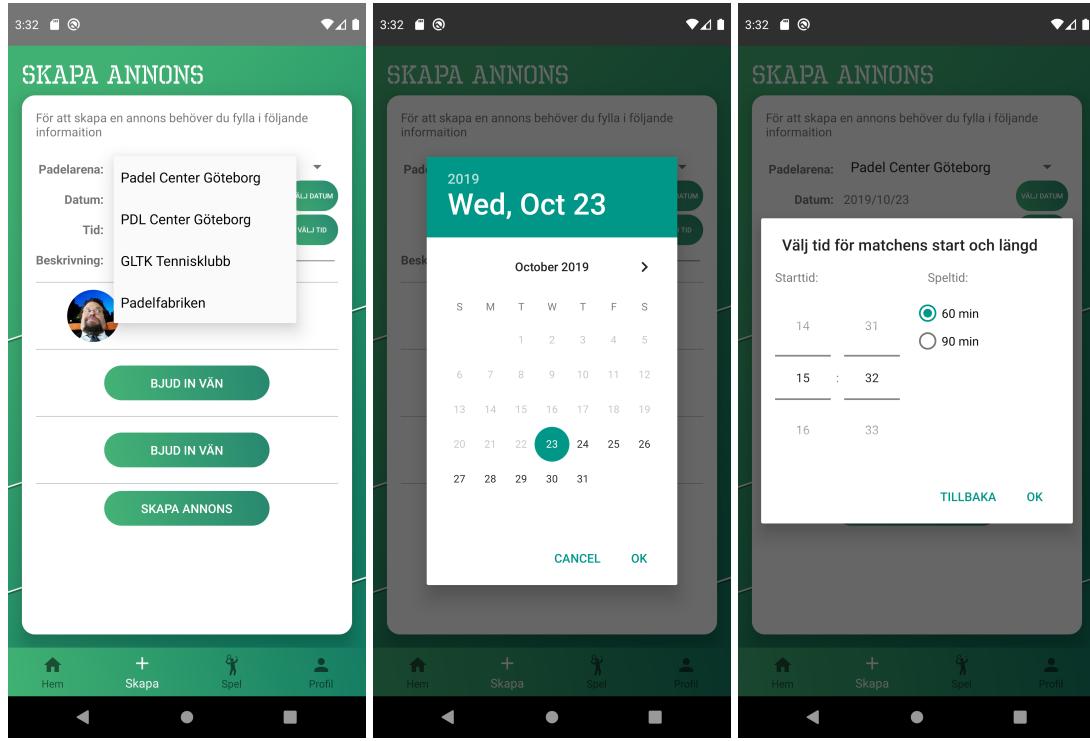
När användaren navigerar sig till vyn där man kan skapa annonser, se Figur 6a, så liknar den skisserna väldigt mycket. Däremot har inte någon vy för att lägga till vänner i annonsen implementerats än, utan klickar man på knappen "Bjud in vän", så läggs en hårdkodad användare till i annonsen, se Figur 6b.



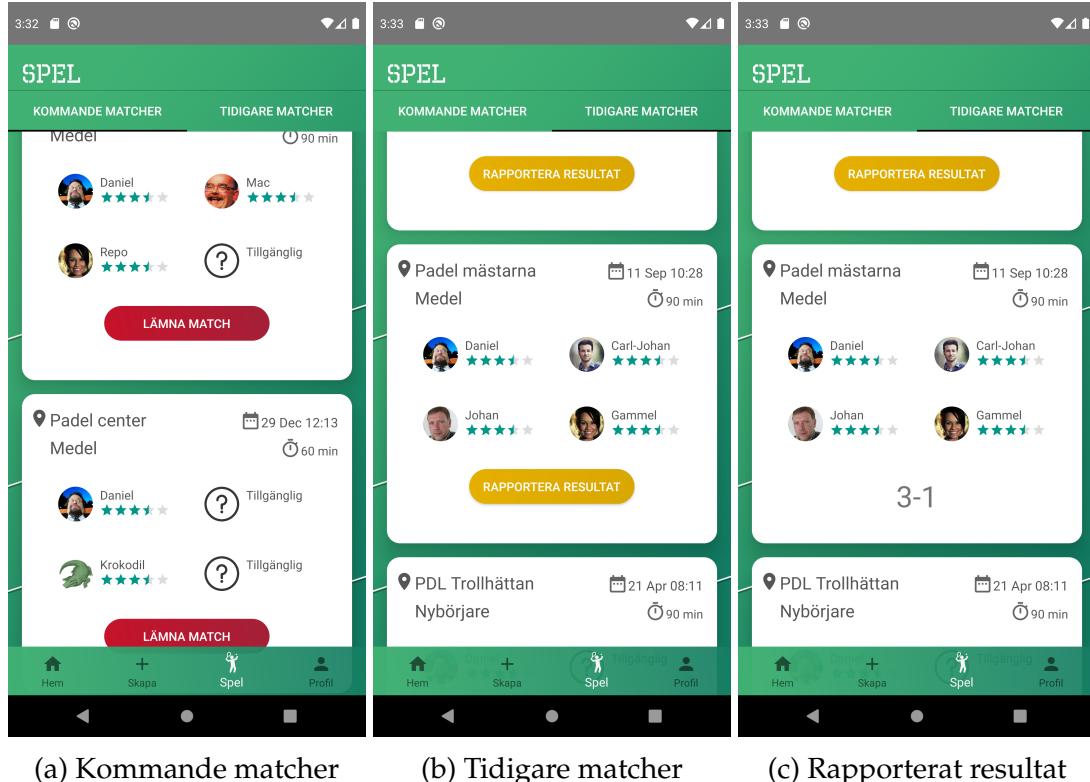
(a) Skapa matchannonser

(b) En inbjuden vän

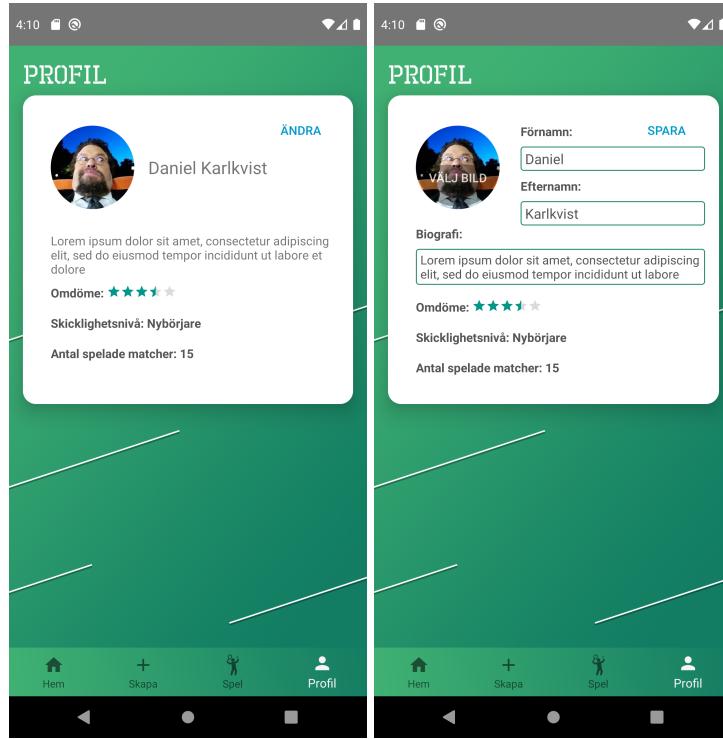
När användaren ska fylla i information kring en annons har hjälpverktyg implementerats. Vilken padelarena som matchen ska hållas i får man välja i en "drop down" meny där några förslag finns att välja mellan, se Figur 7a. Vilket datum som matchen ska spelas får man välja i en kalender vy, se Figur 7b, där man kan välja datum en månad fram i tiden. Tillsist när man ska välja vilken tid är det två steg som ska slutföras, se Figur 7c först vilken tid matchen ska börja och sedan om matchen ska vara 60 minuter eller 90 minuter.



Vyn för att se kommande- & tidigare matcher har fått en del ändringar sen skisserna. Vyn spel har två olika flikar, se Figur 8a & 8b. Där första vyn, Kommande matcher, visar alla annonser man skapat och gått med i, där man även kan lämna dem om man fått förhinder. Den andra vyn, Tidigare matcher, visar alla matcher som spelats. I den vyn kan man även rapportera resultat efter en match slutförts, se Figur 8c.



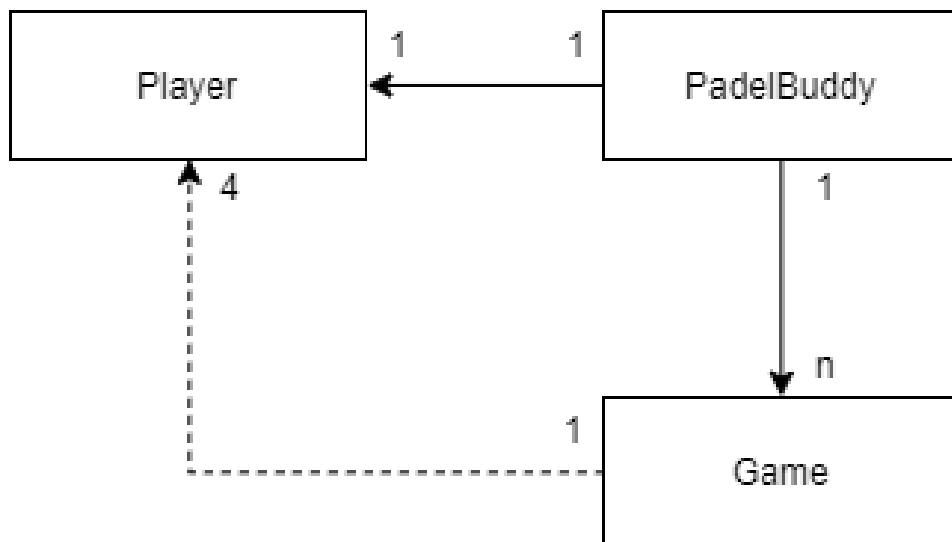
Den implementerade profilvyn, se Figur 9a, är väldigt lik skisserna som gjordes i början av projektet. Däremot finns inte funktionaliteten att skriva kommentarer i aplikationen, vilket är den största grafiska skillnaden.



(a) Din profil

(b) Ändra din profil

3 Domänmodell



(a) Domänmodell

3.1 Klassernas ansvarsområde

3.1.1 PadelBuddy

Klassen PadelBuddy sammankopplar alla andra klasser. Denna klass skapar matchannonser för den inloggade användaren, lägger in & tar bort användaren från matcher och räknar ut vilka annonser som ska visas i de olika vyerna. Den håller även listan med matchannonser.

3.1.2 Player

Klassen Player har ansvaret att hålla i all information om olika spelare, så som namn, skicklighetsnivå, omdöme osv. Det är även denna klass som uppdaterar informationen hos en användare när man redigerar sin profil.

3.1.3 Game

Klassen Game har ansvaret att hålla i all information om en match. När en match skapas så fyller spelaren i var matchen ska spelas, vilket datum och tid, men också hur lång matchen kommer vara. Förutom detta har klassen koll på vilja spelare som gått

med i en matchhannons. Game gör även beräkningar för att exempel räkna ut genomsnittlig skicklighetsnivå, kolla om matchen har slutförts och lägger till nya spelare som väljer att gå med.

4 References

[Android Studio](#)

[Adobe XD](#)

[hdodenhof rounded corners](#)

[Espresso](#)

System design document för Padel Buddy

1. Introduktion

Detta dokument beskriver den fullständiga systemdesignen och arkitekturen för applikationen Padel Buddy. Det är en applikation skapad för Android vars syfte är att förenkla sökandet av motståndare i sporten Padel, samt simplifiera processen vid bestämmelse av plats och tid. Följande kapitel kommer att behandla hur applikationens olika komponenter kommunicerar med varandra, samt redogörelse för implementeringen.

1.1. Definitioner, akronymer och förkortningar

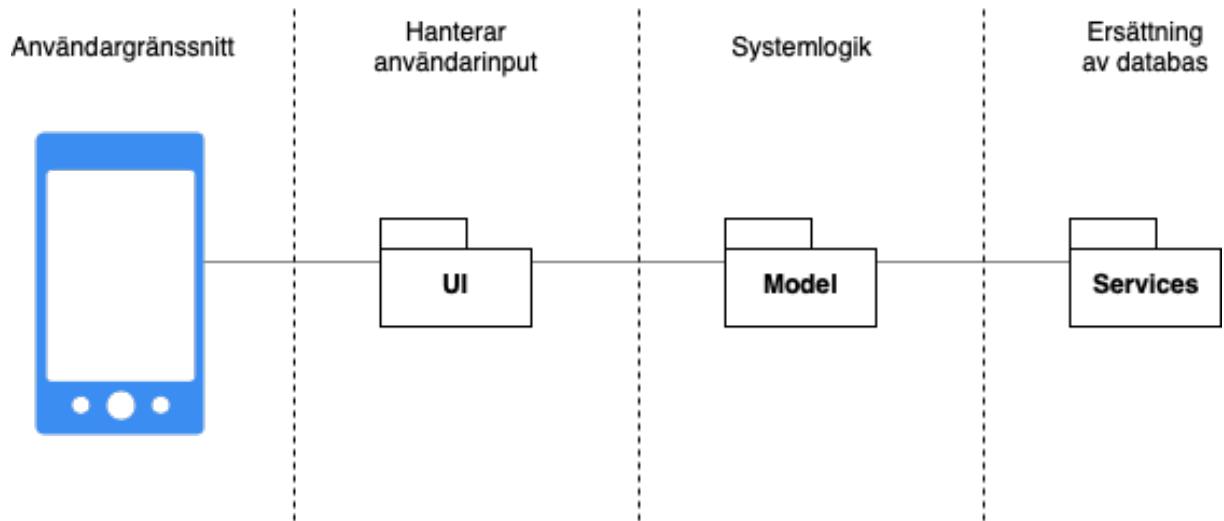
App - Applikation.

Matchannons - Ett kort som finns i "Hem"-fliken och innehåller 4 st. spelare, datum, tid och plats för en match.

Skicklighetsnivå - Ett mått på hur duktig en spelare är.

Omdöme - Ett mått på hur pålitlig en spelare är. Till exempel om spelaren dyker upp i tid.

2. Arkitektur för systemet



Figur 1: Arkitektur för systemet på en hög nivå

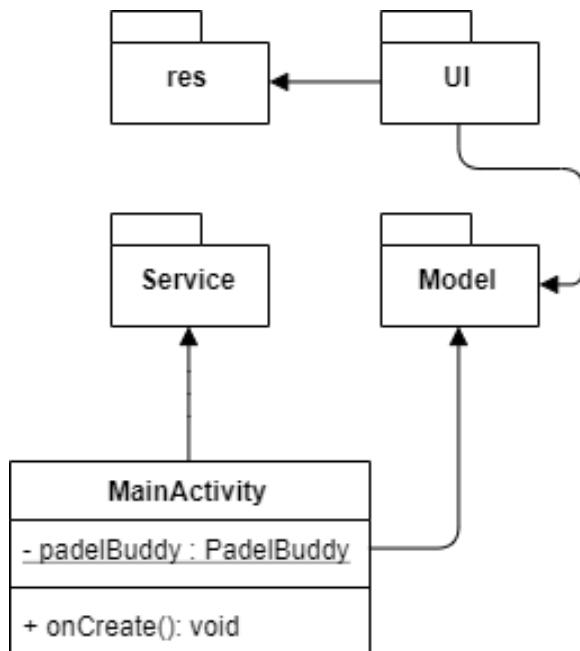
Applikationen är uppdelad i fyra olika delar som visas i figuren. Till vänster är användargränssnittet som representerar allt som en användare faktiskt ser när applikationen används. Användarinputs från när en användare interagerar med gränssnittet registreras av UI-paketet. Detta paket kommunicerar med både gränssnittet och modellen för att utföra de logiska operationerna som förväntas. Model-paketet tillhandahåller applikationens logik och är helt fristående från UI-paketet. Som tillfällig datalagring och generering av testdata används Services-paketet, vilket fungerar som en ersättning av en databas. Modellen hämtar testdata från just Services-paketet istället för från en databas. Ett exempel är när en användare väljer att skapa en annons och trycker på denna knapp. Knappen visas i användargränssnittet och tryckningen registreras av UI-paketet. Paketet kallar då på metoden i modellen som har hand om att skapa en annons.

3. Systemdesign

I följande avsnitt kommer systemets design att både diskuteras och förklaras.

3.1. Översta nivån av alla komponenter

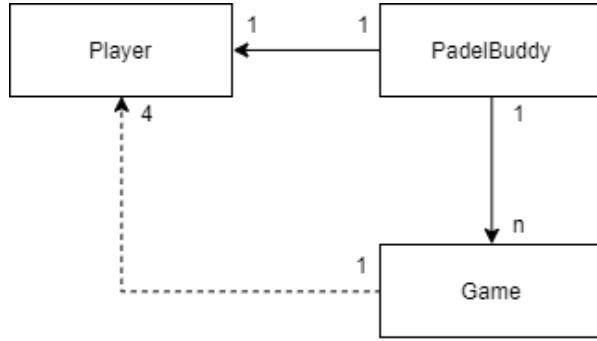
I Figur 2 visas ett UML-diagram över applikationen på paketnivå. I Model används klassen *PadelBuddy* som modellens aggregerade objekt. Denna klass samt gränssnittet *IGame*, *ICreate* och *IPlayer* är de enda klasserna som exponeras utåt. Klassen *PadelBuddy* initieras med användare och matcher från Services i *MainActivity*. Klassen *MainActivity* har en statisk *PadelBuddy*. *MainActivity* har på så vis beroende till både Services och Model.



Figur 2: Arkitektur för systemet på paket-nivå

3.2. Domänmodell

Vår domänmodell består av *Player*, *PadelBuddy* och *Game*. *PadelBuddy* innehåller en *IPlayer*, som i detta fall representeras av en *Player*, och är användaren av applikationen. Tanken är att en användare, genom *PadelBuddy*, ska kunna skapa en annons, vilket i vår applikation kallas för *Game*. Efter att man har skapat en annons ska den synas för andra spelare som använder sig av *PadelBuddy*. Där hanteras alla annonser samt skapa plattformen för att matcha ihop spelare.

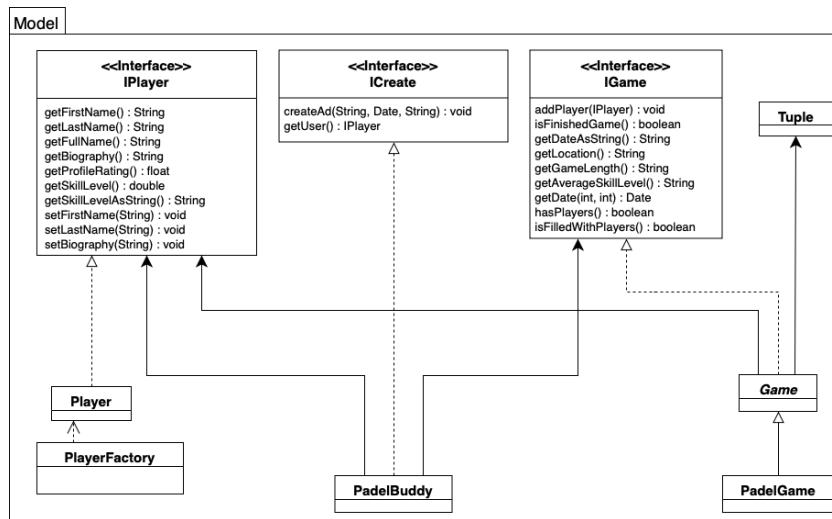


Figur 3: Domänmodell

3.3. Designmodell

Utifrån domänmodellen skapades applikationens designmodell. I designmodellen återfinns samtliga klasser från domänmodellen. Klassen *PadelBuddy* beror på *IPlayer* genom att *PadelBuddy* har en instans av en *IPlayer* i form av en användare. Beroendet till *IGame* uppstår genom att *PadelBuddy* har en lista med *IGame* vilket är applikationens alla olika matchannonser. *IGame* implementeras av *Game*, vilket i sin tur har beroende till *IPlayer* då en match innehåller fyra spelare. En matchannonsskapas med den statiska typen *IGame*, och dynamiska typen *PadelGame*, vilket sätter antalet spelare till fyra.

Inom paketet exponeras våra gränssnitt *IPlayer*, *IGame*, *ICreate* och vårt aggregerade objekt *PadelBuddy*. Resterande klasser är ej åtkomstbara utanför detta paket. Här finns applikationens logik och behandling av data, inget gällande hur data ska visas för användare. Detta går i hand med designmönstret MVC och likaså att modellen inte har beroenden utåt. Modellen blir på så sätt fristående från andra moduler.



Figur 4: Designmodell

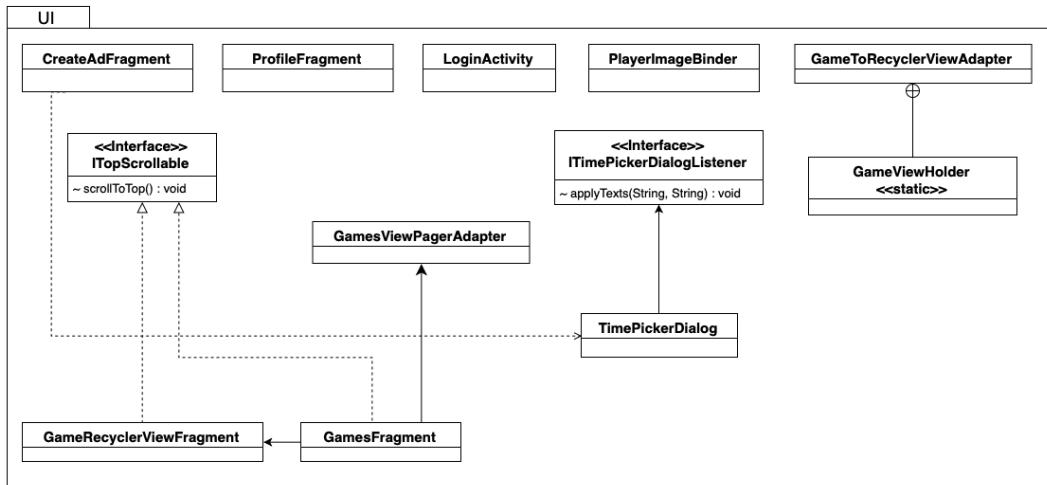
3.4. UI

UI-paketet kommunicerar med aggregatklassen *PadelBuddy* genom klassen *GameToRecyclerViewAdapter*, vilket fungerar som en adapter mellan matcher och en RecyclerView. *GameToRecyclerViewAdapter* innehåller en statisk klass *GameViewHolder* som ansvarar för innehållet av en matchannon. Även klasserna *CreateAdFragment*, *ProfileFragment* samt *GAMESFragment* har ett beroende till *PadelBuddy*. *ProfileFragment* för att komma åt data om användaren och resterande för att få tillgång till listan med *IGame*.

IPlayer och *IGame* är båda gränssnitt som kommunicerar från Model-paketet till UI-paketet. Detta gjordes för att inte *PadelBuddy*-objektet ska bli som en flaskhals, där allt går igenom, vilket även medför till många kedjeanrop. Detta möjliggjorde för oss att endast exponera de metoder som klasser i UI-paketet, som exempelvis *ProfileFragment* behöver. På så vis kunde inkapslingen av den interna logiken optimeras inuti Model-paketet, vilket bidrar till en bättre modulär design.

ITimePickerDialogListener är ett gränssnitt som exponerar en metod vid namn *applyTexts*, som används i klassen *MainActivity*. *LoginActivity* är också en klass som används i *MainActivity*, där *MainActivity* sätter den som startsida för användaren.

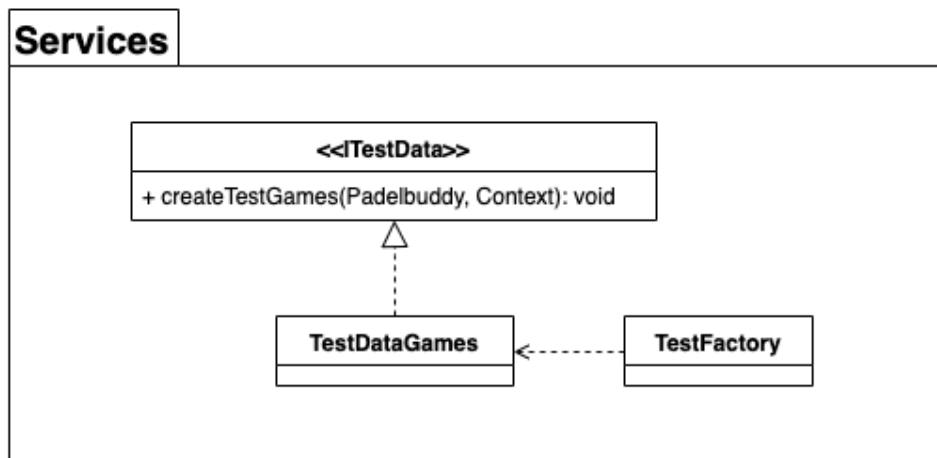
PlayerImageBinder är en klass som innehåller en Map som binder ihop alla spelare och dess motsvarande profilbilder. Den har statiska metoder för att koppla samman en *IPlayer* med en bild samt hämta en spelares bild. Metoderna är statiska för att spelarna och bilderna skall vara samma över hela applikationen.



Figur 5: UI-paketet

3.5. Services

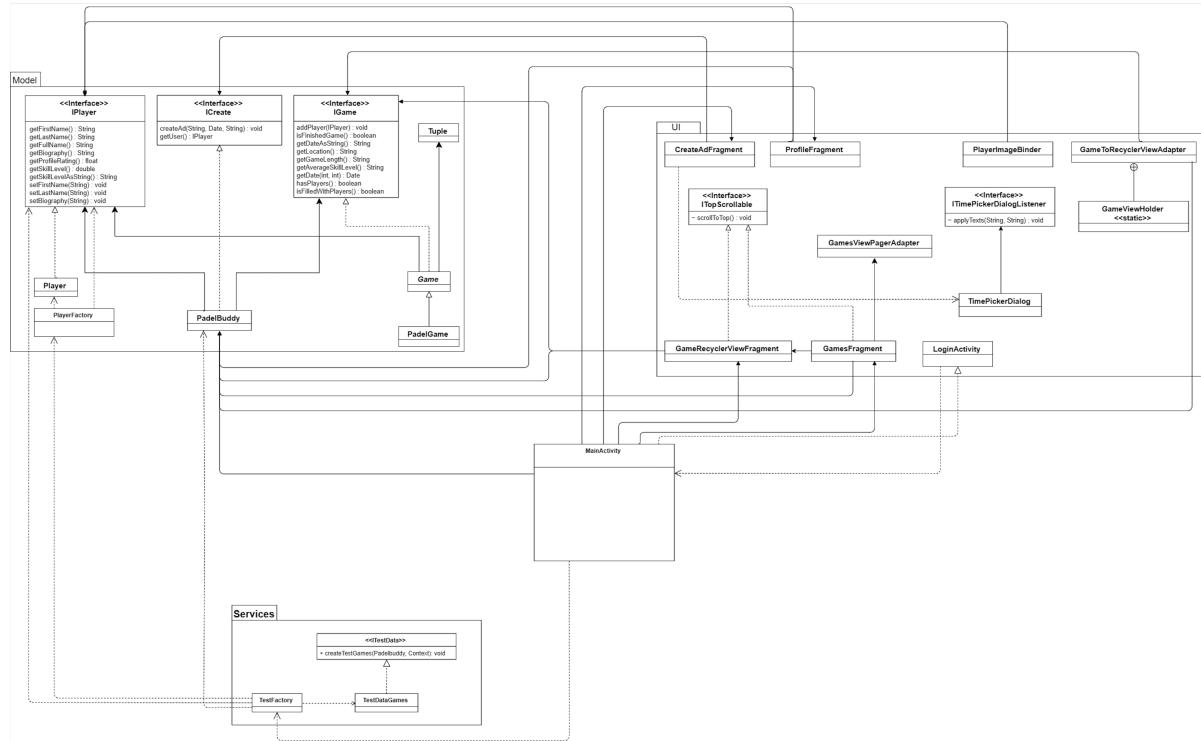
I Services finns klasserna: *TestFactory*, *TestDataGames* och *ITestData*. Paketets ansvarsområde är att agera ersättare för en framtida databas. Istället för att lagra data så skapas här nya testspelare samt testmatcher varje gång applikationen startas för att användas i applikationen. Klassen *TestFactory* exponeras utåt och är den klass som *MainActivity* kommunicerar med för att hämta data när applikationen startas. Eftersom nya spelare och matcher skapas här så är paketet beroende på modellens *IPlayer* samt *PadelBuddy*.



Figur 6: UML-diagram för Services

3.6. UML för hela projektet

Figur 7 visar alla beroenden i hela projektet.



Figur 7: UML-diagram för hela projektet

3.7. Designmönster

Factory - Detta designmönster används vid skapandet av hårdkodade testmatcher och testspelare för att inte skapa dessa objekt i modellen. Detta tillämpades även vid skapandet av *IPlayer*, vilket medför att *Player* är package-private. Denna implementation följer open-closed principle då inga objekt skapas i modellen, och det blir lättare vide ett senare tillfälle att byta ut mot en databas. Det följer även designprincipen dependency inversion principle då *Services*-paketet beror på *PlayerFactory* istället för den konkreta implementationen av *Player*. Det här bidrar till att vår kod får lösare beroenden vilket resulterar i att koden blir mer utbyggbar.

Facade - Vi har även genom vår Factory använt oss av designmönstret Facade då vår *PlayerFactory* gömmer intern logik i modellen, agerar även denna som en fasad. Detta bidrar till Open-closed principen då kodbasen får lägre coupling vid eventuell utbyggnad av applikationen.

Module - Detta designmönster applicerades när klasserna delades upp i paket beroende på dess ansvarsområde, vilket förenklade läsbarheten och möjliggjorde inkapsling av information. Detta stödjer designprincipen Separation of Concern då vi har separerat sådant som på ett abstraktionsplan inte har med varandra att göra. Detta gör att vårt program mer utbyggbar då det möjliggör utvecklare att jobba parallellt i programmet. Det bidrar även till att vårt program får high cohesion och low coupling. Detta minskar programmets modulära komplexitet samt gör programmet lättare att underhålla.

4. Hantering av data

I nuläget används sparade bilder, iconer och strängar som finns i applikationens resource- paket. Vid tillfället sker ingen datahantering av externa filer under körning.

5. Kvalité

- Tester för applikationen
 - 1. GUI + JUnit
 - Här hittas dessa tester:
Padel-Buddy/Padel-Buddy-Project/app/src/androidTest/java/com/danielkarlkvist/padelbuddy/Controller
 - 2. JUnit
 - Här hittas dessa tester:
Padel-Buddy/Padel-Buddy-Project/app/src/test/java/com/danielkarlkvist/padelbuddy/Model
- Kända problem
 - Matchnonsense uppdateras inte vid tryckning på ”Gå med” i ”Hem”-fliken, men den läggs ändå i ”Uppkommande matcher”. Detsamma gäller när användaren ska gå ur en matchnons. Någon typ av dynamic hinting skulle underlätta för användaren att veta om den har blivit tryckt på eller inte.
 - Vid val av tid för match går det nu att välja klockslag för alla dygnets minuter. Detta är ett problem då inga padel-anläggningar erbjuder matchertider som startar exempelvis 13:37. Applikationen bör istället erbjuda matchtider varje halvtimme, vilket är mer förekommande.
 - *MainActivity* borde placeras inuti UI-paketet då denna klassen innehåller fragment tillhörande en del av gränssnittet. Att denna klass även har ansvaret att kommunicera med *TestFactory*, för att skapa matcher och spelare, gör att Single Responsibility bryts. Detta ansvar bör ligga hos en annan klass utanför UI.
- PMD
 - Se bilaga A för PMD rapportering.
- Analys över beroende till bibliotek
 - Se bilaga B för beroende rapportering.

5.1. Åtkomstkontroll och säkerhet

Padel Buddy använder inte funktioner som berör dessa områden (än).

6. Referenser

[Activity](#)

[Fragment](#)

[RecyclerView](#)

[Factory Method](#)

[Facade](#)

[Module](#)

A. PMD

Project: Padel Buddy
Scope: Project 'Padel Buddy'
Profile: Default
Results: Enabled coding rules: 83
• PMD: 83
Problems found: 24
• PMD: 24

Time statistics:

- Started: Fri Oct 25 14:06:17 CEST 2019
- Initialization: 00:00:00.011
- PMD: 00:00:00.707
- Gathering results: 00:00:00.019
- Finished: Fri Oct 25 14:06:17 CEST 2019

Violations:

- Best Practices Count: 24**
 - Avoid Print Stack Trace Count: 1**
 - CreateAdFragment Count: 1
 - Avoid printStackTrace(); use a logger call instead.
 - Preserve Stack Trace Count: 1**
 - TimePickerDialog Count: 1
 - New exception is thrown in catch block, original stack trace may be lost
 - System Printn Count: 1**
 - PlayerImageBinder Count: 1
 - System.out.println is used
 - Unused Private Field Count: 3**
 - Player Count: 3
 - Avoid unused private fields such as 'mail'.
 - Avoid unused private fields such as 'phone'.
 - Avoid unused private fields such as 'age'.
 - Code Style Count: 1**
 - Unnecessary Local Before Return Count: 1
 - TestDataGames Count: 1
 - Consider simply returning the value vs storing it in local variable 'gameDate'
 - Design Count: 11**
 - Final Field Could Be Static Count: 1
 - TestDataGames Count: 1
 - This final field could be made static
 - Singular Field Count: 10
 - CreateAdFragment Count: 3
 - Perhaps 'userImageView' could be replaced by a local variable.
 - Perhaps 'calendar' could be replaced by a local variable.
 - Perhaps 'datePickerDialog' could be replaced by a local variable.
 - GameRecyclerViewFragment Count: 1
 - Perhaps 'gameRecyclerViewAdapter' could be replaced by a local variable.
 - GamesFragment Count: 3
 - Perhaps 'gamesViewPager' could be replaced by a local variable.
 - Perhaps 'upcomingGameFragment' could be replaced by a local variable.
 - Perhaps 'historyGameFragment' could be replaced by a local variable.
 - Player Count: 3
 - Perhaps 'mail' could be replaced by a local variable.
 - Perhaps 'phone' could be replaced by a local variable.
 - Perhaps 'age' could be replaced by a local variable.
- Error Prone Count: 6**
 - Avoid Duplicate Literals Count: 2
 - TestDataGames Count: 1
 - The String literal "lorem" appears 12 times in this file; the first occurrence is on line 112
 - TestFactory Count: 1
 - The String literal "0701234567" appears 4 times in this file; the first occurrence is on line 31
 - Compare Objects With Equals Count: 4
 - MainActivity Count: 2
 - Use equals() to compare object references.
 - Use equals() to compare object references.
 - PadelBuddy Count: 2
 - Use equals() to compare object references.
 - Use equals() to compare object references.

B. Analys över beroende till bibliotek

- ▼  Library Classes (76 entries)
 - ▼  < Android API 28 Platform > (62 entries)
 -  android (2 entries)
 -  android.annotation (1 entry)
 -  android.app (4 entries)
 -  android.content (5 entries)
 -  android.content.pm (1 entry)
 -  android.content.res (1 entry)
 -  android.graphics (2 entries)
 -  android.net (1 entry)
 -  android.os (3 entries)
 -  android.provider (1 entry)
 -  android.text (3 entries)
 -  android.view (3 entries)
 -  android.view.inputmethod (1 entry)
 -  android.widget (14 entries)
 -  java.io (1 entry)
 -  java.lang (9 entries)
 -  java.text (3 entries)
 -  java.util (7 entries)
 -  Gradle: androidx.annotation:annotation:1.0.0@jar (2 entries)
 -  Gradle: androidx.appcompat:appcompat:1.0.2@aar (2 entries)
 -  Gradle: androidx.fragment:fragment:1.0.0@aar (5 entries)
 -  Gradle: androidx.recyclerview:recyclerview:1.0.0@aar (2 entries)
 -  Gradle: androidx.viewpager:viewpager:1.0.0@aar (1 entry)
 -  Gradle: com.google.android.material:material:1.0.0@aar (1 entry)
 -  Gradle: de.hdodenhof:circleimageview:3.0.1@aar (1 entry)

Peer review

1 Do the design and implementation follow design principles?

1.1 SOLID

- S: Modellens klasser har avgränsade ansvarsområden.
- O: Inga abstraktioner eller interfaces används i modellen, vilket är essentiellt för denna princip. Samtliga instansvariabler är dock private, vilket är bra.
- L: Använder inga arv i modellen.
- I: EvaluateOccasionActivity och FullscreenOccasionActivity implementerar GestureListener men lämnar många av metoderna tomta, vilket medför att denna princip ej följs.
- D: Samtliga beroenden till modellen går till konkreta klasser och inte på abstraktioner.

1.2 Command Query

Denna princip följs överlag, men det finns metoder som borde ses över. Ett exempel på detta är `toString()` i Occasion. Bra lösning i TeamUp där det finns flera metoder med samma namn men med olika parameterlistor för att följa Command Query. Ett exempel på detta är `createEmployee()`. Metoden som bryter mot Command Query är väl motiverad då det finns en metod med samma namn som inte bryter mot principen. I Builder-klasserna finns dock setters som bryter mot Command Query.

1.3 Composition over inheritance

Används flitigt, inga arv existerar.

1.4 Separation of Concern

Paketstrukturen för detta projekt är lite oklar. I ui-paketet finns klasser som, enligt oss, kanske tillhör paketet employees inuti admin. Om inte, fel namn på klassen. Likaså gäller för paketet occasions.

1.5 High cohesion and low coupling

Modellen har high cohesion inom paketet för att den beror endast på klasser inom modellen. Det finns även hög coupling på modellen från utomstående paket.

2 Does the project use a consistent coding style?

Bra namn på diverse klasser, men ibland för lika för att avgöra dess ansvarsområde, ex: OccassionAdapter kontra OccasionsAdapter. I Occasion och TeamUp klasserna ligger getters, setters och andra metoder huller om buller. Namn kan variera som i ContractActivity exempelvis, där det finns en Button som heter navigationBtn och under finns addOccasionButton. Inte alltid konsekvent med hur olika element ska döpas i koden.

3 Is the code reusable?

Få abstraktioner. Mycket upprepningar vid användning av Builder-pattern, mer om detta senare.

4 Is the code documented and are proper names used?

Koden innehåller ingen JavaDoc och är överlag inte speciellt dokumenterad. Några metoder innehåller kommentarer för specifika raders ansvar, men inget övergripande för hela metoder. I metoden `toString()` i Occasion hittades dock några variabler som kanske kan förbättra sitt namn, t.ex. `beginsAtDate` och `endsAtDate`. Dessa variabler låter lite som booleans, och kanske kan förbättra sitt namn med t.ex `startDate` och `endDate`. Ibland används även bara en bokstav som o, a och v. Occasion är ett vagt namn för vad klassen egentligen gör och står för. Det är svårt att förstå, från att bara kolla klassens namn, dess ansvarsområde eller syfte.

5 Is the code well tested?

Testerna i projektet är inte riktigt kompletta, då bara tre av klasserna i modellen testas och även dessa har inte 100% coverage. En av de klasserna som testas är Occasion, och de största metoderna testas i denna klass. Däremot innehåller klassen en del getters med mer logik än att bara returnera ett värde, dessa metoder borde också testas. Klassen TeamUp är den andra klassen som har specifika tester, men det är samma visa för den klassen. Allt testas inte och en testerna går inte att köra, eftersom den inte kommer åt metoden `createContract()`. I testerna för TeamUp testas även metoder från klassen Employee, men denna klass borde ha sina egna tester också.

6 Are design patterns used?

Builder Pattern samt Singleton används i dagsläget.

7 Is the design modular? Are there any unnecessary dependencies?

Klassen App är en singleton vilket innebär att många klasser beror på App. Detta innebär att det kan vara svårt att testa. Modellen är uppbyggd utan onödiga beroenden.

8 Does the code use proper abstractions?

OccasionViewHandler är en abstrakt klass som ärvs av både OccasionsAdapter och OccasionActivity.

9 Is the code easy to understand? Is the model isolated from the other parts?

Koden är lätt att förstå, genom att läsa namnet på metoden och sedan följa raderna förstår man vad som ska hända. Modellen är isolerad och har inga beroenden till övriga klasser i projektet.

10 Can the design or code be improved? Are there better solutions?

SchedulingActivitys onCreate()-metod är alldeles för stor och måste refaktoreras, det samma gäller för metoden i ScrollActivity. InstantiateItem()-metoden för klassen OccasionAdapter kan även brytas ner i flera olika metoder då den är ganska lång. Behöver Builder-pattern användas? Vi tycker att det blir onödigt komplicerat, instansvariablerna behöver i princip skrivas på två ställen - kodduplicering.