PROJECT CONNECT 4

Designdokument

VT-15 DA336A, Grupp 23

Kalle Bornemark	861209
Emil Sandgren	941224
Jimmy Maksymiw	891031
Erik Sandgren	941224

20/5 2015 **V 5.0**

Innehållsförteckning

Revisionshistorik

Namn	Datum	Beskrivning	Version
Kalle Bornemark	20/3 2015	Första utgåvan	0.1
Erik Sandgren	22/3 2015	Lade till systemdiagram	0.2
Kalle Bornemark	30/3 2015	Finslipning till v1.0	1.0
Jimmy Maksymiw	1/4 2015	Uppdaterat klassdiagram för servern	1.0.1
Jimmy Maksymiw	9/4 2015	Nytt ER-diagram	1.1
Erik Sandgren	13/4 2015	Uppdatering inför v2.0	2.0
Erik Sandgren	24/4 2015	Uppdaterat klassdiagram och UI-design	3.0
Jimmy Maksymiw	11/5 2015	Uppdaterat ER-diagram	3.3
Erik Sandgren	11/5 2015	Nytt klassdiagram klient/server/Fragment	3.5
Jimmy Maksymiw	11/5 2015	Uppdaterad info om de olika diagrammen.	4.0
Kalle Bornemark	18/5 2015	Lagt till nya diagram och skisser. Uppdaterat alla texter.	4.1
Jimmy Maksymiw	18/5 2015	Bakgrund, syfte, omfattning och ordlista tillagt.	4.2
Erik Sandgren	19/5 2015	Lagt till entity, control och boundary - klasser	4.3
Jimmy Maksymiw, Kalle Bornemark, Erik Sandgren & Emil Sandgren	20/5 2015	Slutförande av v5.0	5.0

Bakgrund

Systemet som ska utvecklas är en applikation till Android och en server skriven i java med tillhörande databas i SQLite.
Applikationen kommer att köras på Android-telefoner där användaren i huvudsyfte spelar det klassiska spelet fyra i rad. Man ska även kunna skapa ett konto för att spela över nätverk mot andra och hålla koll på sin spelarstatistik. Power-ups och varierande spellägen är andra funktioner som kommer finnas.
Serversidan kommer bestå av ett program skrivet i java som kommunicerar med databasen och Android-telefonerna, vilket möjliggör nätverksspel.
En förstudie gjordes innan påbörjandet av detta projekt och det är den som är grunden till detta projekt.

Syfte

Syftet med detta dokument är att ge läsaren en klar bild över systemet och hur de olika delarna hänger ihop och kommunicerar med varandra, såväl som att presentera de skisser vi tagit fram inför formgivningen.

Omfattning

Detta dokument innehåller en övergripande beskrivning över systemet med hjälp av kontextdiagram och systemdiagram.

- Use-case-diagram som visar interaktionen mellan användaren och systemet
- Klassdiagram för klient respektive server-sidan
- ER-diagram för databasen
- Skisser som visar hur det grafiska gränssnittet ut mot användaren är planerat att se ut.

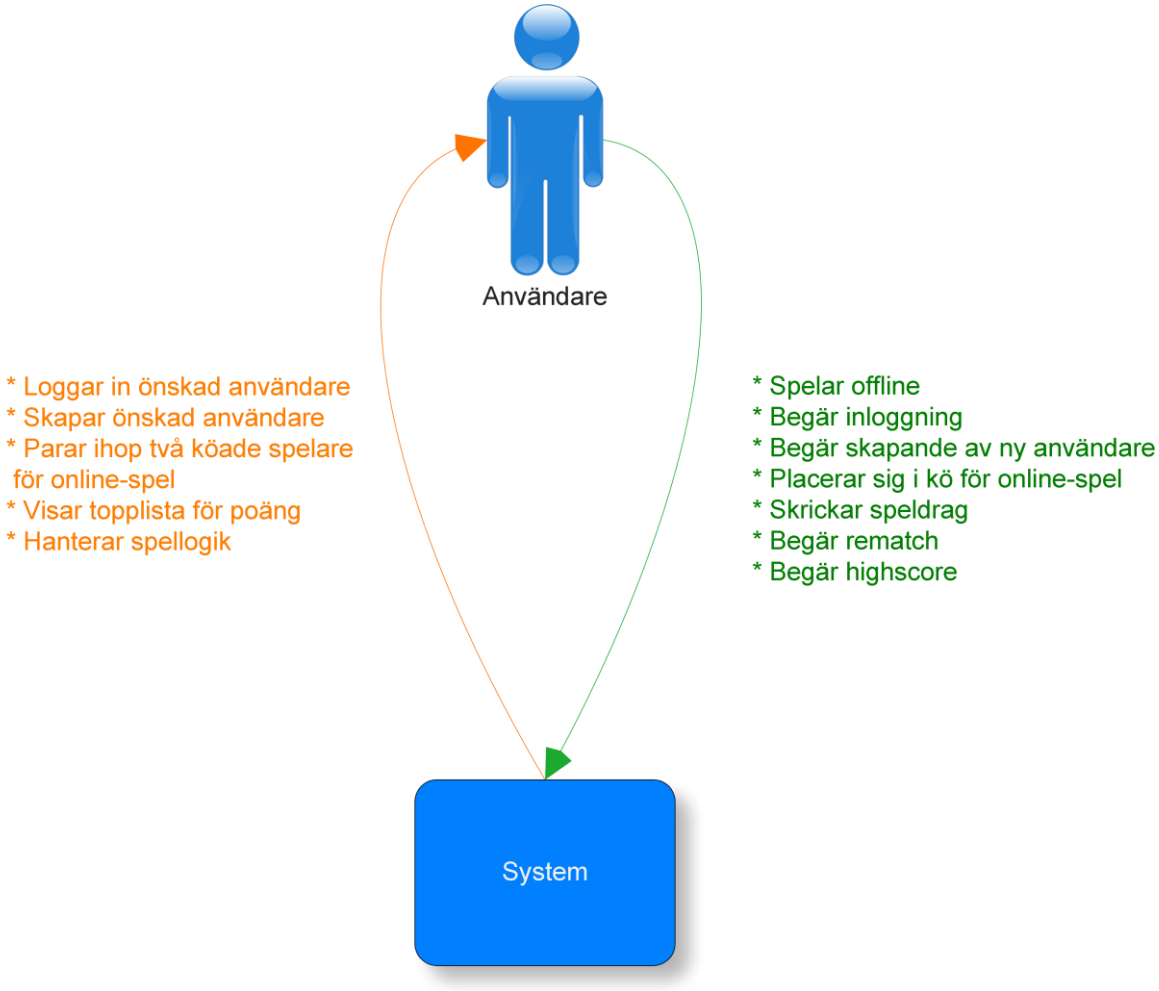
Ordlista

Android – Android är ett öppet mobilt operativsystem för främst smartphones och pekplattor som utvecklas av Google.
Power-up – Ett extra objekt som ger den spelare som tar det en fördel.
Användare – En person som använder applikationen.
Nätverksspel – Ett spel som utförs mellan två användare på varsin enhet över internet.
Lokalt spel – Ett spel som utförs mellan två användare på en enhet.
Elo – En metod för att ranka en spelares relativa styrka.
Highscore – En lista som är baserad på de spelare som har högst elo eller flest vinster/förluster/oavgjort.
Applikation – Det program som är installerat på användarens smartphone som kör operativsystemet Android.
Server – Den dator som kör server-applikationen och har en internet-förbindelse.
Databas – Finns på samma dator som server-applikationen. I databasen är all användarinformation lagrad och hämtas/uppdateras när en användare interagerar med servern.
Systemet – Med systemet menas applikationen, servern och databasen tillsammans.

Kontextdiagram

Beskrivning

Kontextdiagrammet nedan visar vilken grundläggande kommunikation som sker mellan användare och system. Högersidan (grön färg) visar vilka önskemål och navigeringsval användaren gör i applikationen, och vänstersidan (orange färg) visar hur server hanterar detta med gensvar.

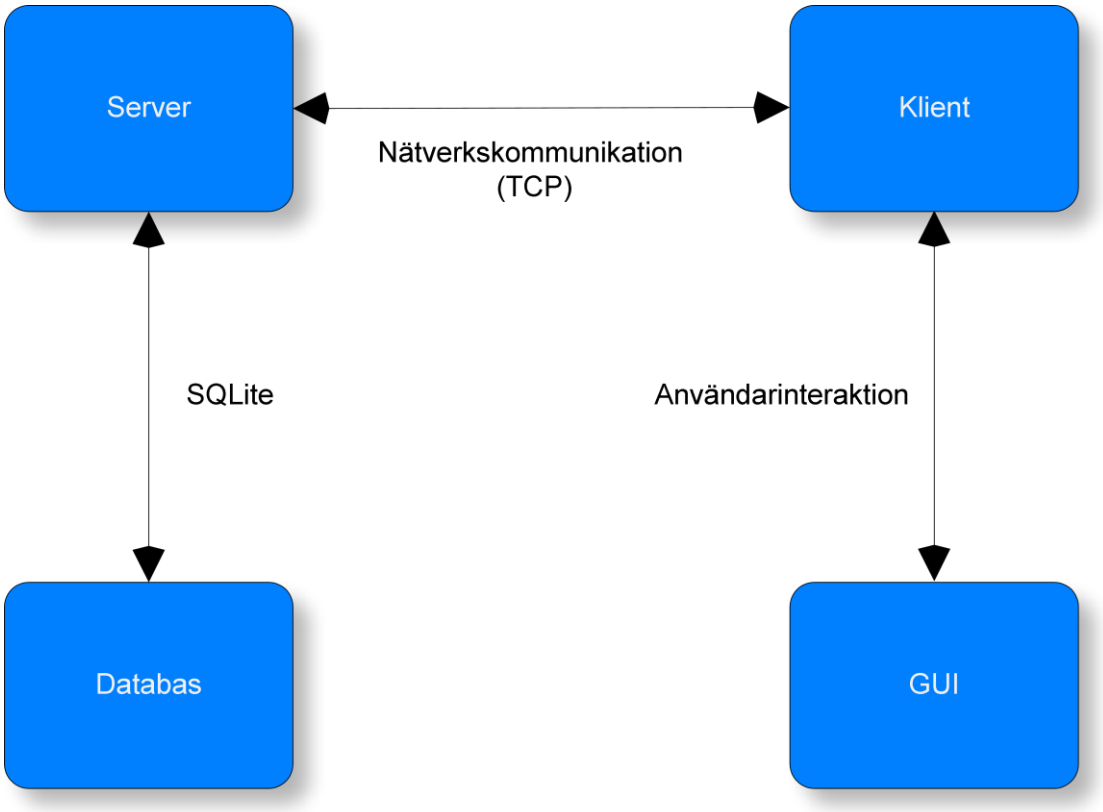


Systemdiagram

Beskrivning

Vi använder oss av ett klient/server/databas-system där varje klient ansluter sig till servern och därmed får tillgång till sitt konto via databasen.

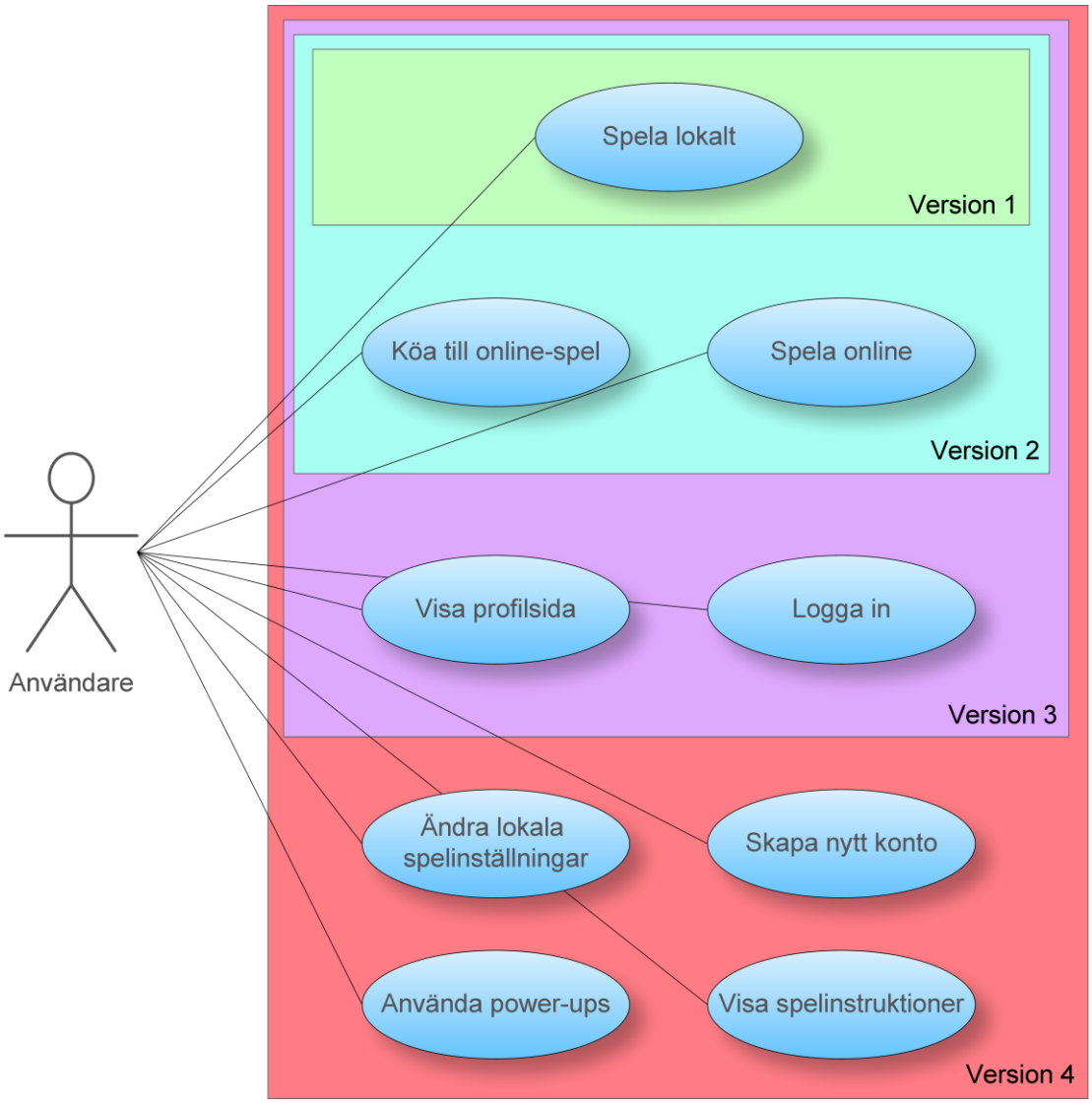
Figuren nedan visar det grundläggande systemupplägget.



Use-case-diagram

Beskrivning

Use case-diagrammet nedan visar vilken funktionalitet användaren har tillgång till i applikationens olika versioner. Detta speglar även den inkrementella utvecklingsprocess vi beskrivit i projektplanen.



Skisser

Beskrivning

Skissen nedan visar den grafiska profil vi skapade i förberedelse för utvecklingen av spelet. Stort fokus lades på en ”platt” och stilren design, samt ambitionen att bibehålla återkommande element på en och samma plats. Ett exempel på detta är loggan som visas i applikationens överkant.

En färgpalett togs fram och användes sedan kontinuerligt i alla designelement för att ge ett genomtänkt och enhetligt intryck.

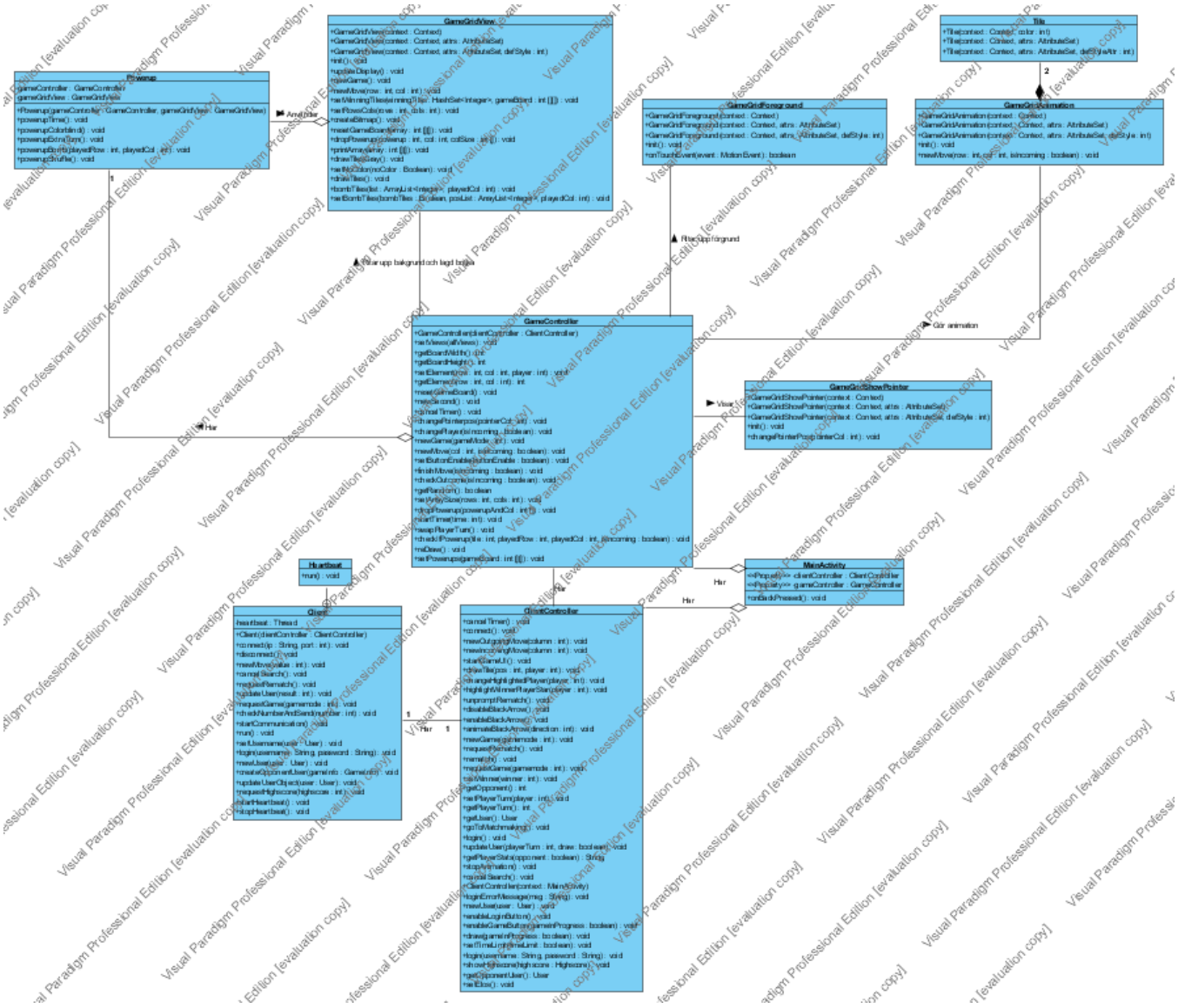
Klassdiagram

Klient

Klassdiagrammet nedan visar strukturen för de klasser som hanterar logiken och de som sköter grafiken för spelbrädet.

- Klassen *ClientController* sköter främst logiken mellan klasserna *Client* och *GameController*
- Klassen *GameController* hanterar all logik av spelet och grafiken av spelbrädet
- Klassen *Client* hanterar kommunikationen mellan applikationen och servern
- Den inre klassen *Heartbeat* ger en säkrare server/klient-anslutning med felhantering
- Entity-klasser: *Client, Tile, Powerup*
- Control-klasser: *ClientController, GameController*
- Boundary-klasser: *GameGridView, GameGridForeground, GameGridAnimation, GameGridShowPointer och alla fragment-klasser.*

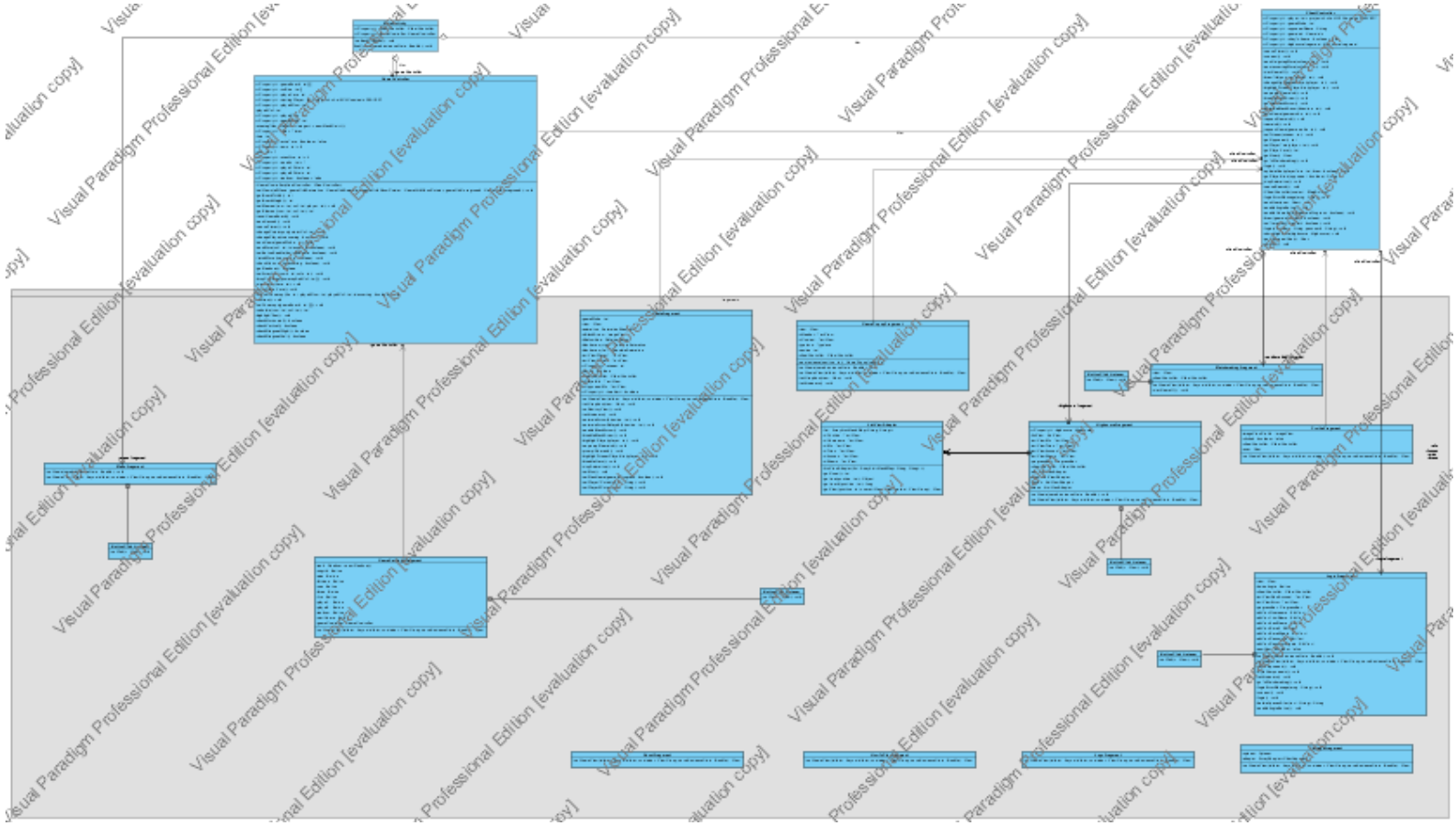
Övriga grafiska komponenter och deras klasstruktur finns under rubriken Fragments på nästkommande sida.



Fragments

Klassdiagrammet nedan visar hur de grafiska komponenterna kommunicerar med varandra.

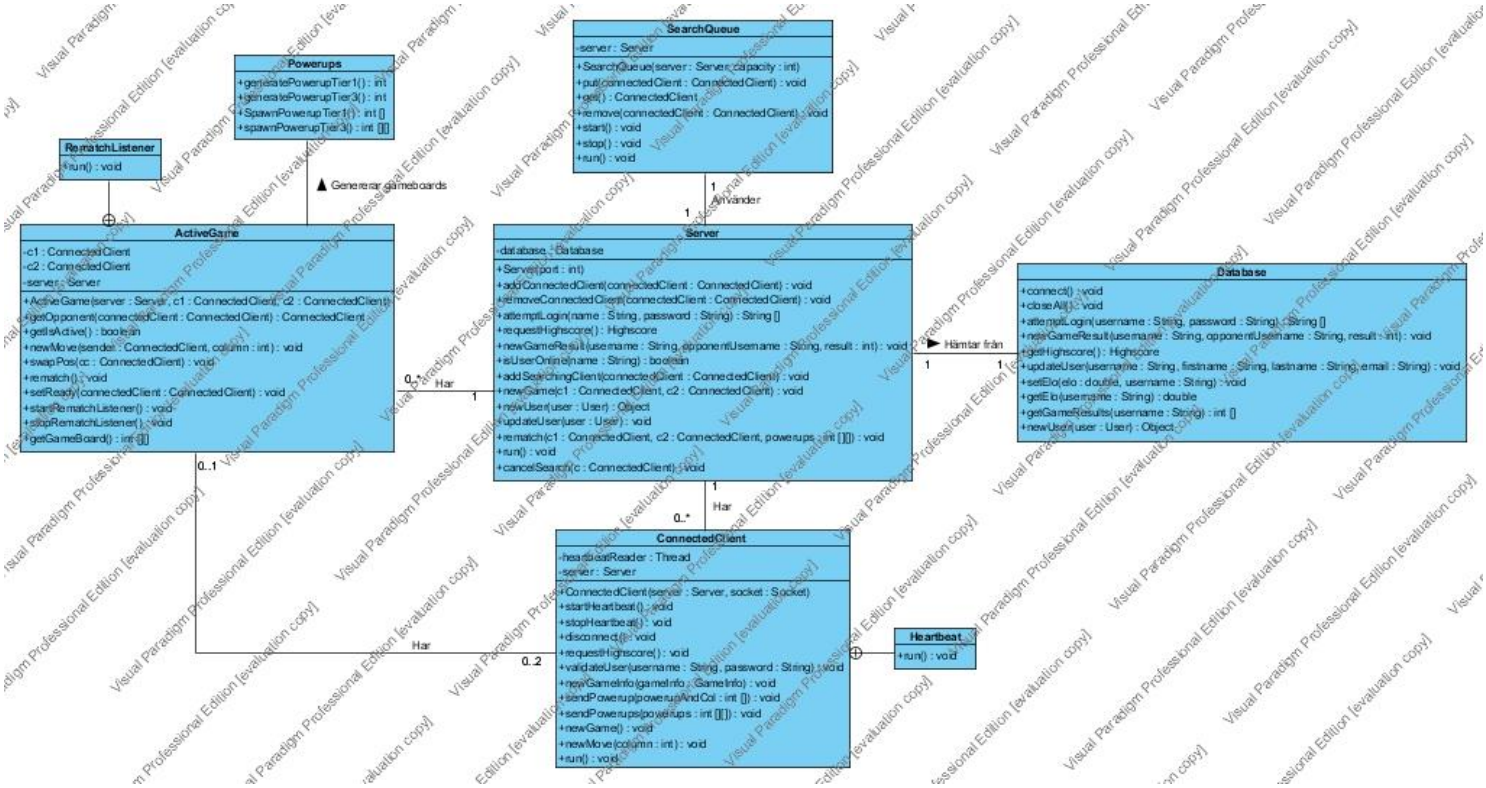
- Det är klassen *ClientController* som hanterar vilket fragment som ska visas beroende på användarens interaktion
- Om det finns ett pågående spel så sköts grafiken genom klassen *GameController*



Server

Klassdiagrammet nedan visar strukturen och kommunikationen mellan klasserna på serversidan.

- Klassen *Server* sköter all kommunikation mellan klasserna *Database* och *ConnectedClient* samt har koll på vilka användare som är anslutna
- Klassen *ConnectedClient* fungerar som mellanhand mellan klienten och servern, och är den koppling som finns mellan två klienter då de spelar nätverksspel mot varandra
- Klassen *Database* används för att komma åt den information som ligger lagrad i databasen
- Klassen *SearchQueue* hanterar de klienter som söker en motspelare att ansluta sig mot för nätverksspel
- Klassen *ActiveGame* hanterar ett pågående spel mellan två klienter och använder sig av klassen *Powerups* för att framkalla power-ups på spelbrädet.
- Klassen *RematchListener* används om klienterna vill spela mot varandra igen efter ett avslutat nätverksspel
- Några entityklasser kan vara: *Powerups, Database, SearchQueue, ConnectedClient, ActiveGame*
- Klassen *Server* kan ses som en kontroller mellan dessa klasser



ER-diagram

Beskrivning

ER-diagrammet nedan visar vilken information vi lagrar för varje unik användare. Databasen innehåller endast en tabell då användarinformation är allt vårt system kräver att sparas.

- Kolumnen *username* används som primärnyckel
- Kolumnerna *username* och *email* måste vara unika för varje användare