CHALMERS

Like To Learn
En frontend- och backendlösning implementerad i Unity och Java

Examensarbete inom Data- och Informationsteknik

JENNY FORSBERG
FIFI JOHANSSON

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2016

Like To Learn
En frontend- och backendlösning implementerad i Unity och Java
Jenny Forsberg
Fifi Johansson

Examinator: Peter Lundin

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
412 96 Göteborg
Telefon: 031-772 1000

Omslag:
Bild från Like To Learns spelvärld

Institutionen för Data- och Informationsteknik
Göteborg 2016

SAMMANFATTNING

Like To Learn är en läroplattform bestående av Unityspel som kan styras av lärare via en webbapplikation. På en webbsida kan en lärare skapa kurser som sedan elever kan träna på genom att spela spelet. Syftet med produkten är att göra lärandet, framför allt läxförhör, roligare för elever, men även att underlätta lärares arbetssituation genom att minska arbetet med rättning.

Uppsatsen syftar till att implementera en klient-server-lösning i ett befintligt Unityspel och undersöka hur väl Java fungerar som en RESTful service. Ramverket Spring har använts för att utveckla programvara. Det stödjer ett enkel säkerhetsskydd inom ramen och motsvarande testmetoder. En prototyp av denna lösning presenteras i en virtuell Linux-maskin som nu är placerad i Azures molnmiljö.

Unityspelet, som används i examensarbetet, skapades ursprungligen i ett annat projekt. Det är ett pedagogiskt 3D-spel riktat till mellanstadieelever, och består av ett blockbyggsystem tillsammans med flera minispel. Under examensarbetet har en klientlösning implementerats i det precis som ett flerspelarläge. Lösningen skapar möjlighet för spelare att följa sina framsteg i spelet på flera olika plattformar.

Eftersom det är mycket viktigt att ta med användarna tidigt under en utvecklingsprocess har en undersökning gjorts för att kontrollera användbarheten hos produkten. Dessa användarrecensioner föreslår olika saker som framtida funktioner, vissa redan genomförda under arbetsgång.

Nyckelord: klient-server, RESTful, Unity, Spring, Java, pedagogik, gamification

ABSTRACT

Like To Learn is an educational platform consisting of a Unity game that can be controlled by the teacher via a web application. On the web page, a teacher can create courses for their students who is supposed to practice these by playing the game. The purpose of this product is to make the small tests, especially the homework tests, more fun for the students. It also facilitates the working conditions for teachers by reducing the work of correction.

This thesis aims to implement a client-server solution into an existing Unity game and investigate the compatibility of Java as the RESTful service. The framework Spring has been used for the developing purpose. Meanwhile, it also provides a simple security protection within the framework as well as the corresponding test methods. A prototype of this solution is presented in a virtual Linux machine placed in Azure's cloud environment.

The Unity game, which is used in the thesis, was originally created in another project. It is an educational 3D game for primary school students, and consists of a block building system along with several minigames. During the thesis work, a client solution has been implemented as the multiplayer mode of this game. The solution allows players to track their game progress across multiple platforms.

Since it is very important to involve the users during the developing process, if the product is supposed to be useable, a survey has been done to check the usability. These user reviews proposed various ideas for possible future features, some already implemented.

Keywords: client-server, RESTful, Unity, Spring, Java, education, gamification

FÖRORD

Semcon är det företag som gjort det möjligt för oss att utveckla vårt ide om ett digitalt läroverktyg för grundskolan till en verklig produkt. Därför vill vi börja med att säga vårt tack till dem för deras stora engagemang och vilja att kreativt stötta oss under vårt arbete. Vi vill särskilt tacka Anna Funke, Mattias Almljung och Mikael Wiktorell för all deras hjälp som våra handledare.

Vi vill också tacka alla de lärare på Chalmers tekniska högskola som på allt sätt uppmuntrat oss.

Det är tack vara dessa som vi idag kan presentera vår produkt *Like To Learn*.

FÖRKORTNINGSLISTA

API - Application Programming Interface, ett applikationsprogrammeringsgränssnitt som kan användas för att kommunicera med en specifik programvara.

DAO - Data Access Object, ett designmönster för att splittra mellan SQL-förfrågningar och annan programmeringskod.

EJB - Enterprise JavaBeans, en Java Enterprice Edition komponent som erbjuder tjänster för bland annat transaktion och säkerhet.

JAR - Java Archive, paketeringsformat för Javaklasser.

JSON - JavaScript Object Notation, ett textbaserat format som används för att utbyta data.

JSP - Java Server Page, teknologi för kontroll av i webbsidor genom användandet av servlets.

MVC - Model-View-Controller, ett designmönster som användas främst för att separera Model paket (data) och View paket (presentation) i applikationen.

POJO - Plain Old Java Objects, ett objekt i Java som implementerar få interfaces och ärver flera superklasser.

REST - Representational State Transfer, en mjukvaruarkitektur för World Wide Web.

URI - Uniform Resource Identifier, en sträng som används för att identifiera en viss resurs. Den vanligaste formen är Uniform Resource Locator (URL).

ORDFÖRKLARING

Back-end - Begrepp som används inom informationsteknik för att beteckna själva basbearbetningen, oftast på servernivå.

Dependency Injection - Ett designmönster i Spring som håller alla klasser så självständiga som möjligt i ett program.

Front-end - Begrepp som används inom informationsteknik för att beteckna den bearbetning som sker av eller nära användaren, d.v.s. användargränssnittsorienterad bearbetning.

Gamification - Spelifiering, användandet av spelmekanismer inom verksamheter som traditionellt inte hör samman med spelande - t.ex. handel, IT eller utbildning.

Klient-server - Ett designmönster som kännetecknas av att olika programvarukomponenter kommunicerar via ett tydligt gränssnitt. Den ena komponenten, klienten, begär uttryckligen tjänster av den andra, servern.

.NET ramverk - Ett programmeringsramverk av Microsoft som stödjer flera olika programmeringsspråk i samma utvecklingsmiljö.

.NET språk - Språk som stödjer .NET ramverk är C#, C++, Visual Basic .NET, J# och Borland Developer Studio (Delphi 2005 och 2006).

REST API - Ett webservice-API, definierat för att ha en bas-URI, en mediatyp för internet (t.ex. JSON) samt HTTP:s standardmetoder (OPTIONS, GET, PUT, POST och DELETE).

Salt - En slumpvald sträng som har lagts till lösenordet för att höja säkerheten.

Servlet - Små program som körs på webbservern för att ändra webbapplikationen innan den skickas iväg till användaren som sökte den.

xUnit-arkitektur - Ett samlat namn för den gemensamma arkitektur och funktionalitet som finns i flera enhetstestramverk, bl.a. JUnit och NUnit.

Table of Contents

SAMMANFATTNING ii
ABSTRACT iii
FÖRORD iv
FÖRKORTNINGSLISTA v
ORDFÖRKLARING vi
1. Inledning 1
1.1 Bakgrund 1
1.2. Mål 2
1.3. Syfte 2
1.4. Avgränsningar 2
1.5. Metod 2
2. Teknisk bakgrund 4
2.1 Verktyg 4
2.1.1. Spelmotorn Unity 4
2.1.2. Utvecklingsmiljön Visual Studio 4
2.1.3. Databashanteraren PostgreSQL 4
2.1.4. Utvecklingsmiljön Eclipse 4
2.1.5. Molnplattformen Azure 4
2.1.6. Versionshanteringssystemet Git 4
2.1.7. Webbtjänsten Google Drive 5
2.1.8. Byggsystemet Maven 5
2.1.9. Applikationsservern Tomcat 5
2.1.10. Utvecklingsmiljön Netbeans 5
2.1.11. Linuxdistrubutionen Ubuntu 5
2.2 Programmeringsspråk 5
2.2.1. Java 5
2.2.2. C# 5
2.2.3. SQL 5
2.3 Ramverk 6
2.3.1. Webbramverket Spring 6
2.3.2. Testramverket JUnit 6
2.3.3. Testramverket NUnit 6
3. Metod 7
3.1 Användaren 7
3.1.1 Användarfall 7
3.1.2 Undersökning bland olika typer av användare 8

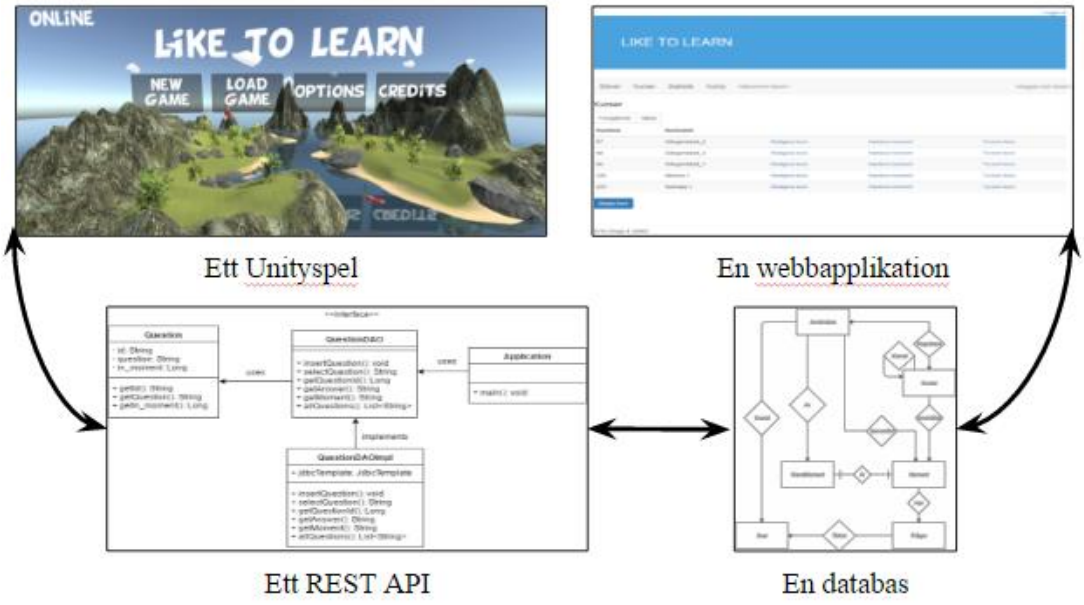
3.2. Molnlösning 9
3.3. Presentation av statistik 9
4. Klienten 11
4.1. Serveranrop 11
4.2. Online/Offline-mode 12
4.3. Test i Unityklienten 13
5. Server 14
5.1. REST API 14
5.2. Designmönster 15
5.3. Test i REST API 16
6. Databas 18
7. Resultat 19
7.1 Serverlösningen med ett REST API i Java 19
7.2 Implementationen av en klientlösning i Unity 19
7.3 Förbättringen av databasen 19
7.4 Övrigt 20
8. Diskussion och slutsats 20
8.1 Hållbar utveckling 20
8.2 Serverlösningen i Java 20
8.3 Klientlösningen i Unity 21
8.4 Databasen 21
8.5 Användaren 21
8.6 Kritisk diskussion 22
8.7 Slutsatser 23
9. Framtida utveckling 24
9.1 Framtida användare 24
10. Referenser 26
11. Bilagor 28
11.1. Bilaga 1: Bilder från Unityspelet 28
11.2. Bilaga 2: Undersökningsenkäten 29
11.3. Bilaga 3: Lärarundersökning 30
11.4. Bilaga 4: Föräldrarundersökning 32

1. Inledning

Detta kapitel handlar om hur det här arbetet påbörjades samt dess syfte, mål och avgränsningar.

1.1 Bakgrund

Like To Learn består i dag av fyra olika delar, ett spel gjort i Unity, en webbapplikation, ett REST API och en databas. Unityspelet, webbapplikationen och databasen har utvecklats under tidigare kurser i utbildningen[4][5]. Under det här arbetet har den fjärde delen utvecklats, nämligen REST API:n, samt all kommunikation med den.



Figur 1.1 Like To Learns olika delar. Pilarna symboliserar delarnas kommunikation.

Unityspelet utvecklades under projektkursen DAT065 VT 16 [4]. Det ett pedagogiskt spel, utvecklat för att kunna användas i en utbildningsmiljö som till exempel grundskolans mellanstadium. Tanken är att det som förr var ensidig och tråkig repetition nu ska kunna vändas till en rolig lek. Spelet innehåller tre minispel. I dem kan eleverna öva på de frågor som läraren skapat samtidigt som de utmanar sig själva i olika små aktiviteter. När de lyckats med sina uppgifter får de små belöningar. Dessa kan man sedan använda för att bygga och skapa egna saker, såsom byggnader, borgar eller hela världar.

Ett annat syfte med spelet är att avlasta läraren genom att minska arbetsbördan med rättning. Spelet är självrättande och håller reda på vilka frågor en elev klarat eller behöver öva mer på.

Webbapplikationen är skapad i web-applikationskursen DAT076, även den under vt 16 [5]. Syftet med den är att lärare lätt ska kunna skapa kurser till sina elever. Kurserna innehåller ett eller flera moment med tillhörande frågor. Dessa kan sedan elever ansluta sig till genom att fylla i rätt kurskod.

Databasen skapades samtidigt som webbapplikationen i samma kurs. Dess uppgift är att hålla reda på användare och kurser så att det lätt går att ta reda på vem som läser vad och hur det går för varje enskild elev.

När REST API:n skulle skapas visade sig konsultföretaget Semcon vara intresserade av de tekniker som API:n skulle byggas i. Semcon är ett internationellt företag inom branschen för ingenjörstjänster och produktinformation. De utvecklar och formger olika typer av produkter åt många företag inom t.ex. fordons-, energi-, och verkstadsindustrin.

Tillsammans med dem bestämdes det att en arkitektur för en klient- server lösning skulle tas fram. Den skulle kunna användas i många sammanhang, inte bara för grundskolans värld. För att illustrera dess möjligheter visas den här i Like To Learn. Den kan även ses i andra domäner med liknande intressebilder, t.ex. i vården med dess uppföljning av patienter i hemmet.

1.2. Mål

- Utveckla en serverlösning med ett REST API i Java.
- Implementera en klintlösning i det befintliga Unityspelet.
- Förbättra databasen så att den även kan hantera användares svar.
- Tillföra statistik på webbsidan som visar hur användarna svarat på frågorna.
- Lägga till en föräldravy i webbapplikationen som visar elevers svarstatistik.
- Göra en användarundersökning som testar om produkten kan anses vara användarvänlig.

1.3. Syfte

Implementera och testa en klient- och serverlösning i Java och Unity.

1.4. Avgränsningar

Arbetet bearbetar de delar av produkten som berör backend, d.v.s. dess uppbyggnad, dess kommunikation med klienter, hur dess data presenteras samt huruvida detta samverkar till en användarvänlig produkt eller inte. När det gäller frontend används redan befintliga lösningar, ett Unityspel [4] (Se bilaga 11.1) och en webbapplikation [5].

1.5. Metod

Under arbetet ska en funktionell backendlösning utvecklas i Java. Den ska innehålla ett fungerande REST API som kan skriva och läsa information från en databas. En klient som kan kommunicera med detta REST API ska implementeras i det redan befintliga Unityspelet [4]. Vidare ska statistik från den nämnda databasen presenteras på den sedan tidigare utvecklade webbapplikationen [5].

De huvudsakliga programmeringsspråken som kommer användas under det här arbetet är C# och Java, men även SQL behövs. För att underlätta utvecklingsprocessen används även Javaramverket Spring.

För att ta reda på om produkten blir användarvänlig eller inte genomförs en användarstudie parallellt med utvecklingen. Fyra olika användare, två lärare och två föräldrar, väljs slumpmässigt ut och ombeds svara på en enkät.

2. Teknisk bakgrund

För att kunna skapa den önskade användarvänliga produkten krävdes en hel del verktyg och olika tekniker. Nedan beskrivs all teknik som användes i proprietär.

2.1 Verktyg

2.1.1. Spelmotorn Unity

Unity är en populär spelmotor som kan användas till att utveckla spel både 2D- och 3D. [21] Den kan även användas för visualisering inom andra områden än spel, t.ex. inom industrin vilket Semcon har exempel på. I detta arbete har verktyget använts kostnadsfritt.

Spel utvecklade i Unity kan enkelt porteras till olika plattformar, t.ex. Android, IOS, PC och Mac. På så sätt skapas möjligheten att porta klientlösningen till flera olika mobila och stationära enheter.

2.1.2. Utvecklingsmiljön Visual Studio

Visual Studio är en utvecklingsmiljö som är utvecklad av Microsoft för kodning i t.ex. C#. [22] IDE används i kombination med spelmotorn Unity för att utveckla klientlösning.

2.1.3. Databashanteraren PostgreSQL

PostgreSQL är en relationsdatabas som finns som fri programvara [16]. Den används som databas i Like To Learn. Både REST API:n och webbapplikationen läser och skriver i den.

2.1.4. Utvecklingsmiljön Eclipse

Eclipse är en utvecklingsmiljö som är fri programvara. Eclipse kan användas för att skriva kod i många olika programmeringsspråk [19]. Under utvecklingen av projektet har det använts för att utveckla ett REST API i programmeringsspråket Java.

2.1.5. Molnplattformen Azure

Azure är en flexibel molnplattform som har stöd för flera olika operativsystem, programmeringsspråk, ramverk, databaser m.m. [1]. För att göra Like To Learn åtkomligt för allmänheten placerades REST API:n, databasen samt webbapplikationen i Azure. En virtuell Linux-miljö skapades som kunde innehålla allt detta.

2.1.6. Versionshanteringssystemet Git

Versionshanteringssystemet Git är ett system för att versionshantera källkoden under utvecklingen. För att synkronisera källkoden har webbtjänsten Github använts. Git är ett kraftfullt verktyg för samarbete i team som arbetar med källkod. [7]

Verktyget har gjort projektet väldigt lätt att flytta mellan olika plattformar, såsom till exempel från datorn det utvecklats på till Azure i molnet. Gits avancerade lösningar har gjort det lätt för Like To Learns utvecklare att dela kod med varandra.

2.1.7. Webbtjänsten Google Drive

För dokumentation och utbyte av information mellan utvecklare har Google Drive använts. Det är ett verktyg där man lätt kan komma åt sina filer samt redigera dem tillsammans med sina kollegor. [6]

2.1.8. Byggsystemet Maven

Maven är ett automatiserat byggsystem som på ett standaliserat sätt gör att alla steg för att kompilera, länka och packa en applikation [11]. I det här projektet används det framförallt använts i uppbyggnaden av REST API:n.

2.1.9. Applikationsservern Tomcat

Tomcat är en applikationsserver från Apache Software Foundation. Den exekverar Java-servlets och renderar webbsidor som innehåller JSP-kod. [17] I projektet har Tomcat fått som uppgift att vara serverlösning i molnet.

2.1.10. Utvecklingsmiljön Netbeans

Netbeans är en fri utvecklingsmiljö för mjukvaruutvecklare, lämplig för användare som vill utveckla i Java [13]. I projektet används Netbeans främst inom utveckling av webbapplikationen.

2.1.11. Linuxdistrubutionen Ubuntu

Ubuntu är ett Linuxbaserat operativsystem, en plattform fri att använda, dela och utveckla. [20] Plattformen stödjer mängder av fria öppen-källkod program, bl.a. postgreSQL, Spring m.m.

2.2 Programmeringsspråk

2.2.1. Java

Java är ett välanvänt programmeringsspråk som har funnits ända sedan 1995. Eftersom det har använts av så många, i mängder av olika situationer, t.ex. för att utveckla REST API:n, är det väl testat och möjligt att använda till mycket [8].

2.2.2. C#

C# är ett programmeringsspråk som Unity är välanpassat för, implementerades klientlösningen i det. Språket är objektorienterat och har många likheter med Java [2].

2.2.3. SQL

SQL är ett programspråk som utvecklades på 1970-talet för att användas till databaser. Det kan användas för att hämta och ändra på data i en relationsorienterad databashanterare, så som t.ex. PostgreSQL [12]. Under projektets gång har SQL

använts inbäddat i Javakod. Sådan källkod använder både REST API:n och webbapplikationen när de läser och skriver i Like To Learns databas.

2.3 Ramverk

2.3.1. Webbramverket Spring

Spring är ett välanvänt ramverk inom Javaprogrammering. Det är skapat av Rod Johnson och släpptes i första versionen i juni 2003 som öppen programvara. Ramverket uppnådde stor framgång inom kort tid, främst inom webbutveckling. [18]

Det finns många fördelar med att använda Spring i utvecklingsprocessen. I artikeln "Introducing the Spring Framework" nämnde Rod Johnson följande punkter som kan vara intressanta för globala Javautvecklare: [9]

- *Spring är designat så att applikationer som är byggda med den använder så lite API som möjligt. Många affärsobjekt i Spring är oberoende av ramverk, vilket gör att det blir lätt om man sedan vill byta.*
- *Applikationer som är byggda med Spring är väldigt enkla att enhetstesta eftersom koden som var miljöberoende har flyttat in i ramverket. Genom att använda Javabean-stil POJO:s, det blir lättare att använda Dependency Injection för att tillföra testdata.*
- *Spring kan lösa många problem utan behovet från EJB. Spring kan ge ett alternativ till EJB som är lämpligt för många applikationer. [9]*

En viktig teknik som används i Spring heter Dependency Injection. Det är ett exempel på ett designmönster i Spring som används för att tillgodose klassernas beroende. Ett exempel är då en Javaapplikation skapas, programmerare vill då oftast ha klasserna så oberoende av varandra som möjligt. Tanken med det är att klasserna ska kunna återanvändas vid senare tillfällen. Dependency Injection gör det möjligt att binda alla klasser tillsammans men ändå låta dem vara väldigt oberoende av varandra. Detta gör det även lättare för programmerare att skriva enhetstester till enskilda klasser. [15]

2.3.2. Testramverket JUnit

JUnit är ett ramverk som kan används för att skriva enhetstester. Det är ett exempel på en xUnit-arkitektur baserad på programmeringsspråket Java. Motsvarande testramverk finns även i C, C# och Python m.m. I detta arbete användas JUnit tillsammans med Springs egna testmetoder för att utföra omfattande tester på REST API. [10]

2.3.3. Testramverket NUnit

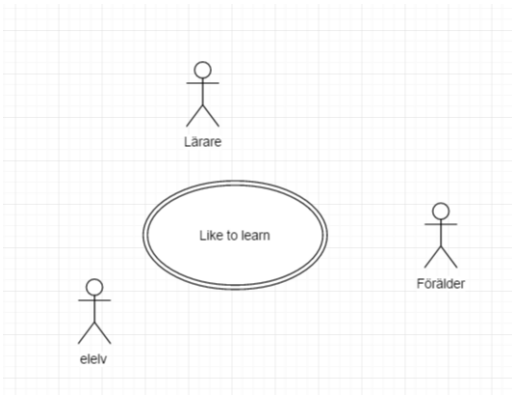
NUnit är ett annat ramverk för enhetstest som är byggt på den ovannämnda xUnit-arkitekturen. Den kan användas i alla .Net språk och har samma funktionalitet som JUnit har i Java. I version 3.0 har ramverket omskrivits så att det innehåller många nya funktioner som är specifika för just .Net språk. [14] När Like To Learn utvecklades används NUnit för att testa klientlösningen.

3. Metod

Det här kapitlet handlar om den använda metoden under examensarbetet.

3.1 Användaren

När man arbetar med gamification står användaren i starkt fokus. Arbetet började därför med att identifiera olika typer av användare samt deras behov.



Figur 2:1 En produkt har olika typer av användare med olika behov. Här visas en av arbetets första skisser på exempelanvändare.

3.1.1 Användarfall

För att tydliggöra vad som var behoven hos de olika användartyperna av vår produkt, identifierades tidigt olika användarfall. Nedan följer en lista av dessa.

- Som lärare vill jag kunna se resultatstatistik från alla mina elever för att veta hur mycket de har förstått samt vad jag behöver förklara för dem.
- Som lärare vill jag kunna skapa en kurs, som handlar om det jag håller på att undervisa om, för att innehållet i spelet ska vara relevant för mig och mina elever.
- Som lärare vill jag kunna redigera en kurs genom att lägga till och ta bort frågor. Jag vill på så sätt se till att innehållet i de kurser jag har skapat alltid är relevant och uppdaterat.
- Som lärare vill jag kunna ta bort en kurs om den inte längre är relevant för mina elever.
- Som lärare vill jag bestämma vilken kurs som ska köras just nu. Detta för att mina elever inte ska byta till någon annan kurs än den som de ska öva på just nu.
- Som lärare vill jag ha rätt att redigera mina elevers uppgifter när eleverna glömmer bort sina lösenord.
- Som lärare vill jag kunna ta bort en elev från en kurs om han/hon inte längre behöver öva på den, har bytt klass eller liknande.
- Som lärare vill jag kunna redigera mina uppgifter för att alltid hålla uppgifterna uppdaterade och byta lösenord regelbundet.
- Som förälder vill jag se resultat från alla mina barn för att hålla mig uppdaterad och se vad de behöver hjälp med.

- Som förälder vill jag ha rätt att redigera mina barns uppgifter när de glömmer bort sina lösenord.
- Som förälder vill jag kunna redigera mina uppgifter för att alltid hålla uppgifterna uppdaterade och byta lösenord regelbundet.
- Som elev vill jag ansluta till mina föräldrar för att visa dem hur det går för mig i skolan.
- Som elev vill jag kunna anmäla mig till en existerande kurs när jag börjar läsa den.
- Som elev vill jag kunna se mina resultat för att veta hur det går för mig.
- Som elev vill jag kunna redigera mina uppgifter för att alltid hålla mina uppgifter uppdaterade och byta mitt lösenord regelbundet.

3.1.2 Undersökning bland olika typer av användare

För att kunna utveckla en användbar produkt är det viktigt att användarna finns med under hela utvecklingsprocessen. På grund av detta har olika typer av användare tagit del av produkten under arbetets gång. De elever som använder Like To Learn kommer huvudsakligen att använda sig av Unityspelet. Därför togs elever med redan under en tidigare del av produktutvecklingen då spelet utvecklades.

Under det här arbetets gång har produkten anpassats för fler typer av användare, nämligen förädlar och lärare. För att få en fingervisning om produkten är funktionell för dem eller inte, gjordes en enkätundersökning. Fyra slumpvis utvalda användare, två lärare och två föräldrar, fick ta del av vår produkt. De fick möjlighet att testa den samt svara på några frågor.

3.1.2.1 Elever

En grupp elever från en mellanstadieklass fick lov och testa produkten och ge sina synpunkter under Like To Learns tidiga utveckling[4]. De tyckte i huvudsak att den del av produkten som är avsedd för dem, d.v.s. Unityspelet, fungerade väl. Någon av dem påpekade dock att det måste vara spelmässigt lätt att ge rätt svar på frågorna i spelet om det ska fungera som ett läxförhör.

3.1.2.2 Lärare

Två lärare tog sig tid att svara på en enkätundersökning om Like To Learn. (Se bilaga 11.2 och 11.3) Svaren visar att båda lärarna ser potentiella användningsområden för produkten. Den skulle kunna minska tiden som läggs på rättning samt ge en tydlig översikt över elevernas resultat.

Önskemål om att kunna se procentsatser för frekvensen av rätt svar fråga för fråga, samt elev för elev framkom. Som lärare är det intressant att veta om det finns en fråga som många svarar fel på, eller om en elev alltid ger samma felsvar på en viss fråga. Ett annat förslag som framkom under undersökningen var att få en tidsuppgift om hur lång tid det tog för varje enskild elev att klara ett visst moment. Det skulle ge en indikation på hur lätt eller svårt uppgiften var för eleverna.

Det uttrycktes också önskemål om att få resultaten sammanställda klassvis, samt om att kunna bestämma lägsta godkända nivå. Frågan var om det alltid måste krävas 100 % rätt för att momentet ska rapporteras som avklarat.

Vissa svagheter hos Like To Learn lyfts fram. En av dem är att vissa felsvar hos eleverna kan bero på svårigheter i spelet istället för svårighet med frågan. Statistiken kan därför bli missvisande. En annan svaghet är att en elev lätt kan skapa ett lärarkonto, som också kan bli ett administrationskonto. Det kan sedan användas för att skapa "mobbingkurser" eller likande.

3.1.2.3 Föräldrar

På samma sätt som lärarna tog sig två föräldrar sig tid att svara på enkäten.(Se bilaga 11.2 och 11.4) De svarade på samma frågor, fast ur ett annat perspektiv.

I deras svar framkom att även föräldrar kan se områden där Like To Learn kan vara användbart. Det finns situationer där produkten skulle kunna underlätta deras vardag, t.ex. när det gäller läxläsning och informationsintag om barnens kunskapsinhämtning.

Olika önskemål framkom även här, så som till exempel att se utvecklingskurvor över barnens resultat samt feedback från läraren.

Inte lika många svagheter med spelet framkom hos dem som i lärarnas svar, endast en om att barnen kan bli sittandes för länge vid spelet.

3.2. Molnlösning

För att göra Like To Learn lättåtkomligt för användare var det viktigt att dess server fanns på ett pålitligt och lättillgängligt ställe. Tanken är att användare när som helst och var som helst ska kunna nå både REST API:n och webbapplikationen. Detta ska gälla så länge det går att upprätta en åtminstone tillfällig internetanslutning. För att göra detta möjligt placerades dessa två delar, samt databasen, i Azures molnlösning.

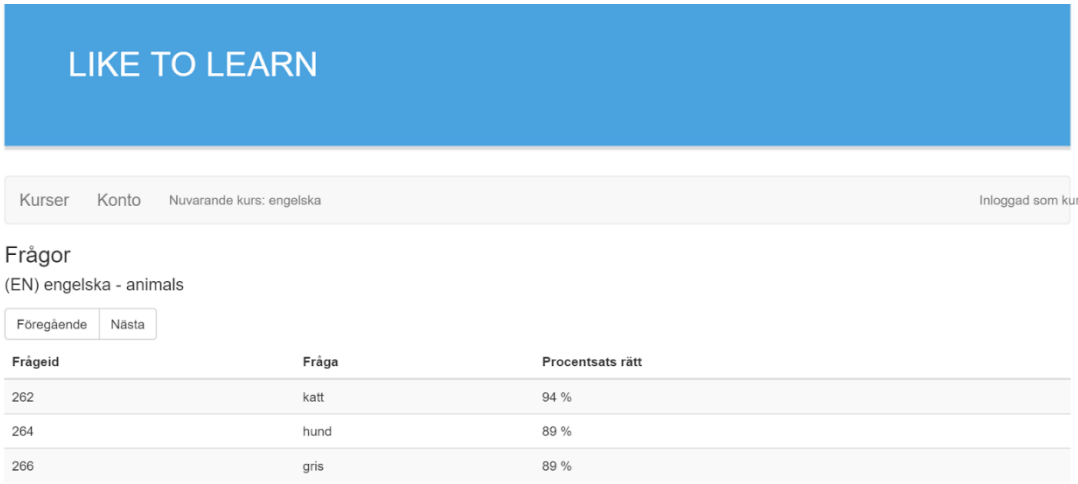
3.2.1. Virtuell maskin

För att enkelt kunna använda databasen som ett gränssnitt mellan webbapplikationen och REST API:n valdes att placera alla tre komponenterna i en virtuell maskin som skapades i molnet. Den virtuella maskinen har fått Linuxdistrubutionen Ubuntu som operativsystem.

3.3. Presentation av statistik

För att all den information om spelares svar och avklarade moment som samlas in ska komma till nytta, måste den göras tillgänglig för användarna. I Like To Learns fall valdes att presentera svarsstatistik bredvid varje kurs, moment samt fråga på den redan befintliga webbapplikationen.

Nedan syns ett exempel på en elevs svarsstatistik på frågor från momentet Animals från kursen Engelska. Här syns att eleven har ganska lätt för de tre frågorna som finns i det här momentet. På ett sätt som liknar statistiken för varje fråga kan användaren också se statistik för varje moment samt varje kurs. För att eleverna ska kunna visa sina resultat även för föräldrar har en extra typ av konton skapats under det här arbetets gång, föräldrakonton. Från ett föräldrakonto kan användaren se statistik från de elever som väljer att visa henne/honom sina resultat.



Figur 3.3 En exempelvy från webbapplikationen som visar ett exempel på hur statistik från spelet presenteras.

I användarstudien framkom önskemål från lärare att få statistik både för varje elev samt för varje moment och fråga i en kurs. Detta har tillgodosetts genom att lärare kan klicka in på varje elev för att se hur många moment som är avklarade. Vidare kan läraren klicka för att se procentsatser rätt svar fråga för fråga. Både varje enskild elevs procentsatser går att se, samt hela gruppens procentsatser. Läraren får även statistik över hur många elever som klarat varje enskilt moment på den sida som momenten skapas.

4. Klienten

Klienten är skriven i Unity, i det redan befintliga Unityspelet. Kod har alltså lagts till där för att spelet ska kunna kommunicera med REST API:n. Eftersom övriga spelet var skrivet i C# skrevs också alla REST-anropen i C#.

För att få spelet att fungera i så många miljöer som möjligt har både ett offline- och ett onlinemode skapats. Om Unityspelet aldrig har haft kontakt med internet kan spelet köras på samma sätt som det gjorde innan klientlösningen integrerades. Då körs spelet i det som kallas testmode. Om man däremot skapar ett nytt spel i onlinemode, börjar klienten kommunicera med REST API:n.

4.1. Serveranrop

Det första som händer när spelet startas online är att en första förfrågan till REST API:n skickas automatiskt. Den innehåller ett hårdkodat användarnamn och lösenord för att öka säkerheten. Meningen är att bara de Unityklienter som har Like To Learns kod ska kunna skicka det här lösenordet och därmed få tillgång till REST API:n.

Den första automatiserade förfrågan frågar efter ett accesstoken m.h.a. koden som visas i figur 4.1.

```
public void authentication()
{
    if (online)                                                     //Kontroller görs att spelet har kontakt med internet
    {
        WWWForm form = new WWWForm();
        form.AddField("username", "jlong");                         // Hårdkodat användarnamn
        form.AddField("password", "chalmers2016!");                 // Hårdkodat lösenord
        form.AddField("grant_type", "password");
        byte[] rawData = form.data;

        string url = string.Format(presentIP + ":8181/oauth/token");    // En url för rätt flik i Rest APIn skapas

                                                                    // Servernamn och serverlösenord kodas:
        String encoded = System.Convert.ToBase64String(System.Text.Encoding.ASCII.GetBytes("liketolearn-restapi:123456"));

        Dictionary<String, String> headers = new Dictionary<string, string>();
        headers.Add("Authorization", "Basic " + encoded);           // Det kodade serverlösenordet läggs i headers

        WWW www = new WWW(url, rawData, headers);                   // Anropet genomförs
        StartCoroutine(WaitForRequest(www, courseList, 2));         // Unity börjar vänta på svar

    }
}
```

Figur 4.1. Metod Authentication från Receive klass i Unity spel.

Som svar på anropet kommer ett JSONobjekt från REST API:n. I det finns ett accesstoken som m.h.a. en egenhändigt konstruerad parser sparas undan. Det kommer sedan att användas i alla kommande anrop från den här anslutningen. På så sätt är tanken att bara de klienter som har Like To Learns lösenord ska hämta information från REST API:n. Anledningen till att parsern är konstruerad på egen hand är att ingen lämplig parser för ändamålet hittades.

När användaren loggar in skickas ett anrop som liknar det ovanstående kodexemplet till API:n. Det innehåller det hämtade accesstoken samt parametrar med det användarnamn och lösenord som användaren försöker logga in med. Om inloggningsuppgifterna stämmer överens med de uppgifter som finns lagrade i databasen, skickar REST API:n tillbaka ordet "true" i ett JSONobjekt. Efter det startar Unityspelet. Om inloggningsuppgifterna istället visar sig vara felaktiga, kommer ordet "false" som svar. Då händer ingenting i Unityklienten, d.v.s., spelet startar inte.

Om inloggningen lyckas börjar klienten skicka regelbundna förfrågningar till REST API:n. Var och en av förfrågningarna har med sig accesstoken tillsammans med olika parametrar.

I början av varje spel skickas en fråga om någon av spelets kurser ska uppdateras eller bytas ut. Det anropet har med sig användarens id som parameter. REST API:n svarar då med en kurskod och ett versionsnummer som gäller den här elevens nuvarande kurs. Om spelet redan har en kurs med den koden och med det versionsnummer inläst, sätts den som nuvarande kurs i spelet. Det betyder att just den kursens frågor börjar användas i spelet.

Om det istället är en ny kurskod som kommer, eller en gammal kurskod med ett nytt versionsnummer, gör klienten ett nytt anrop för att hämta just den kursen. Det nya anropet har med sig den aktuella kurskoden som parametrar. Oavsett om spelet hade en gammal version av kursen innan, eller om det handlar om en helt ny kurs, läses hela kursen i fråga in på nytt. På så sätt får spelet den senaste versionen av kursen med alla dess eventuella ändringar.

Kurserna som hämtas är uppdelade i olika moment som vardera innehåller ett obestämt antal frågor och svar. Under spelets gång svarar användaren på frågorna. Alla svar som ges, oavsett rätt eller fel, sparas. Om spelet för tillfället har en fungerande internetanslutning, skickas svaret iväg till REST API:n för att senare sammanställas. Det skickas tillsammans med användarens id, frågans id samt information om svaret var rätt eller fel.

Inte bara information om användares svar skickas iväg, även information om vilka moment i kursen som är avklarade sänds. När en användare har svarat rätt tre gånger på alla frågor i ett visst moment skickas information om att momentet är avklarat till REST API:n. Det görs tillsammans med en tidsuppgift om hur lång tid det tog.

4.2. Online/Offline-mode

I många sammanhang kan det förekomma att klienten ibland förlorar sin internetanslutning under pågående spel, t.ex. i ett klassrum där upp till 30 elever kan samsas om en accesspunkt gjord för ca 10 användare. För att ingen viktig information ska gå förlorad vid sådana tillfällen finns kod i klienten som kan spara undan saker om internetanslutningen försvinner.

Klienten känner regelbundet av om det går att nå internet eller inte. Det görs genom att REST API:ns ip-adress pingas. Om försöket att pinga misslyckas sparas all information om användarens svar och avklarade moment undan. De läggs som

sparade på fil i väntan på en ny internetanslutning. Om så behövs kan informationen skickas iväg först vid en senare inloggning av samma användare.

4.3. Test i Unityklienten

När Unityklienten skulle testas användes NUnit som i mångt och mycket liknar JUnit.

Med hjälp av det testades vad som händer när information tas emot i klienten. Eftersom information från REST API:n kommer i form av JSONobjekt, skapades sådana på “konstgjord” väg i testen. Dessa skickas in i systemen för olika tester. Ramverket innehåller olika metoder som kan användas för att se om resultatet blev det förväntade eller inte.

Nedan syns ett kodexempel, gjort för att testa att klienten inte skapar en ny kurs om det kommer ett nytt JSONobjekt med en redan känd kurskod.

```
[Test]
public void TestNumberOfCourses()
{
    // Ett jsonobjekt skapas på “konstgjord” väg:
    string jsonString =
        "[{\"coursecode\":\"ENG\",\"momentcode\":\"260\",\"questionid\":\"262\",\"question\":\"katt\",\"answer\":\" cat \"}]";

    // Ett nytt jsonobjekt med samma kurskod, men inte samma fråga, skapas
    string jsonString2 =
        "[{\"coursecode\":\"ENG\",\"momentcode\":\"260\",\"questionid\":\"264\",\"question\":\"hund\",\"answer\":\"dog\"}]";

    parser = new Parser(jsonString, courses, 2);     // Första objektet skickas in för bearbetning
    parser = new Parser(jsonString2, courses, 2);    // Andra objektet skickas in på samma sätt

    Assert.AreEqual(courses.Count, 1);               // Förväntat resultat är att kurslistan "courses" har en kurs
}
```

Figur 4.3. Metod TestNumberOfCourses i testklass SampleTests. Test försäkrar att ingen kurs har blivit tillagda två gånger med samma kurskod.

5. Server

5.1. REST API

REST API är en Javaserverlösning som är utvecklad med Spring som ramverk. Den har som uppgift att hantera dataöverföringen mellan databasen och Unityspelet. Programmet fungerar även som en kommunikationsbrygga mellan olika användare. Därför är det viktigt att skydda användarnas uppgifter, dvs. lösenord och användarnamn, speciellt när man sänder data över internet. Ramverket Spring gör detta enkelt genom användningen av Spring Security.

Säkerhetsgranskningen görs i två steg. I steg ett skickar REST API:n ut ett accesstoken, från adress/oauth/token, efter att den har mottagit korrekta inloggningsuppgifter från ett Unityspel. När detta är gjort väntar REST API:n på att få en ny förfrågan från samma klient. När den kommer hämtas och skickas information från databasen om och endast om Unityspelet har med sig rätt accesstoken. Ett sådant accesstoken måste vara med i alla övriga anrop från Unityklienter om de ska få tillgång till information.

Förfrågningar från Unityspel hanteras därefter i REST APIs API-kontroller. I kontrollern mappas flikar till passande namn beroende på dess funktionaliteter. Sex flikar är skapade i kontrollern för att hantera de allmänna uppgifterna i REST API:n. (Se nedanstående punktlista)

- /statistics - Mottar användares svar från Unityspel och sparar det i databasen.
- /time - Mottar tidsuppgifter om hur lång tid det tog för en spelare att klara av ett befintligt moment. Ett nytt avklarat moment sparas i databasen och lagras med tiden som instansvariabel.
- /version - Sänder versionsnumret och kurskoden av användares nuvarande kurs till Unityspel. Om versionen är uppdaterad ska en ny förfrågan komma från samma Unityklient till questions-fliken.
- /questions - Skickar alla frågor i den nuvarande kursen som användaren är registrerad på. Anrop av fliken kräver ett fungerade användarid från ett Unityspel.
- /login - Mottar användarid och lösenord från Unityspel. Metoden kontrollerar i databasen om användaruppgifterna är korrekta. En sträng som visar om uppgifterna var godkända eller inte returneras.
- /test - Används under utvecklingsprocessen för att testa kontrollfunktion i REST API:n.

Figuren nedan visar ett exempel på en flik i kontroller, questions-fliken. Kommunikation mellan databasen och REST API sker genom dess DAO klass. REST API:n har använt designmönster DAO för att splittra SQL-förfrågningar från den övriga Javakoden.

```java
// Send all questions from database to Unity.
@RequestMapping(value = "/questions")
public List<QuestionJson> sendQuestions(@RequestParam(value = "userid", defaultValue = "0") String userId) {
    String coursecode = courseDAO.currentCourse(userId);
    QuestionJson tmp;
    List<QuestionJson> listTmp = new ArrayList<QuestionJson>();
    for (String moment : momentDAO.allMoments(coursecode)) {
        for (String question : questionDAO.allQuestions(moment)) {
            tmp = new QuestionJson(coursecode, moment, question, questionDAO.selectQuestion(question),
                    questionDAO.getAnswer(question));
            listTmp.add(tmp);
        }
    }
    return listTmp;
}
```

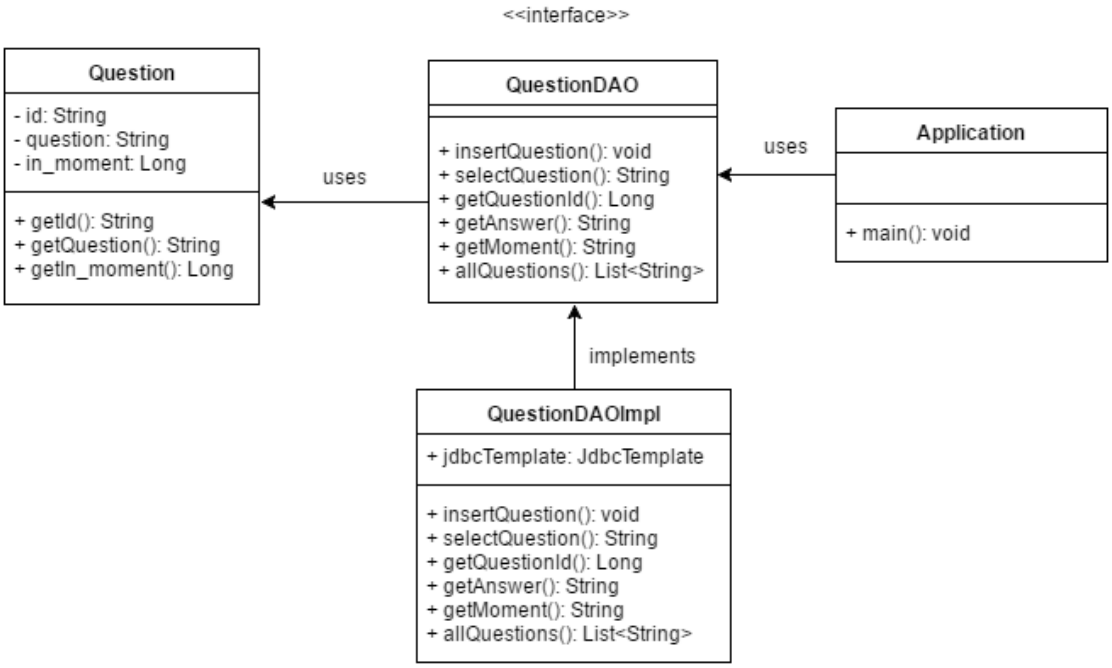
Figur 5.1. Metod SendQuestions i REST APIs kontroller klass. Metod mappas till flik /questions och har funktion till att sänder alla frågor i den nuvarande kursen som användare är registrerad på.

När metoden har körts sparas resultatet i form av ett JSONobjekt. Detta förs över som en dataöverföring till en Unityklient. Det är gjort så för att Unityspel ska kunna hämta information på samma sätt som REST API:n har sparat den. Kontrollern i REST API körs kontinuerligt för att upptäcka förfrågningar så fort de har kommit in. Statistiken från spelet uppdateras därför i realtid, vilket innebär att en lärare kan titta på elevens statistik samtidigt som han/hon spelar Like To Learn.

5.2. Designmönster

I designen av detta projekt användes flera designmönster. I REST API:n till exempel, där är alla klasser skapade i ett klassiskt MVC-mönster. API:n uppdelas därmed i tre huvudsakliga paket: model, view och kontroller. Applikationen använder även ett annat standardformat av designmönster, nämligen Data Access Object (DAO).

Designmönstret DAO implementerades i REST API:n i samband med ramverket Spring. Det har till uppgift att fungera som en mellanhand mellan databasen och REST API:n. Det är tack vare DAO som det bildas en klar och tydlig gränsdragning, så att SQL-förfrågningarna inte blandas med REST-kontrollerna som ligger i en annan klass.



Figur 5.2. UML- diagram av en exempelimplementation i REST API som har utvecklas med designmönstret DAO.

5.3. Test i REST API

För att säkerhetsställa funktionaliteterna i REST API:n har flera enhetstester skrivits under arbetets gång. Testerna använder båda JUnit och Springs standardtestmetoder [18] för att bättre anpassa sig efter Spring-ramverket. En viktig funktion i REST API är att sända och motta accesstoken mellan Unityspelet och REST API:n. I testen skapas en låtsatsförfrågning till REST API med rätt hårdkodade servernamn och lösenord. Koden liknar Unitys API-förfrågningar, men följer naturligtvis en hel annan struktur. Se mer i figur 5.3.

```
@Test
public void passwordGrant() {
    MultiValueMap<String, String> request = new LinkedMultiValueMap<String, String>();
    request.set("username", "jlong");
    request.set("password", "chalmers2016!");
    request.set("grant_type", "password");
    @SuppressWarnings("unchecked")
    Map<String, Object> token = new TestRestTemplate("liketolearn-restapi", "123456")
            .postForObject("http://localhost:" + port + "/oauth/token", request,
                    Map.class);
    assertNotNull("Wrong response: " + token, token.get("access_token"));
}
```

Figur 5.3. Metod PasswordGrant i testklass ApplicationTests. Test försäkrar att REST API skickar tillbaka rätt accesstoken efter den har mottagit en hårdkodad login.

Metoden passwordGrant garanterar, med hjälp av rätt hårdkodade login, att REST API skickar tillbaka rätt accesstoken. Testet assertNotNull testar om token blir null. Om en inte-null accesstoken har mottagits passerar det, annars misslyckas det.

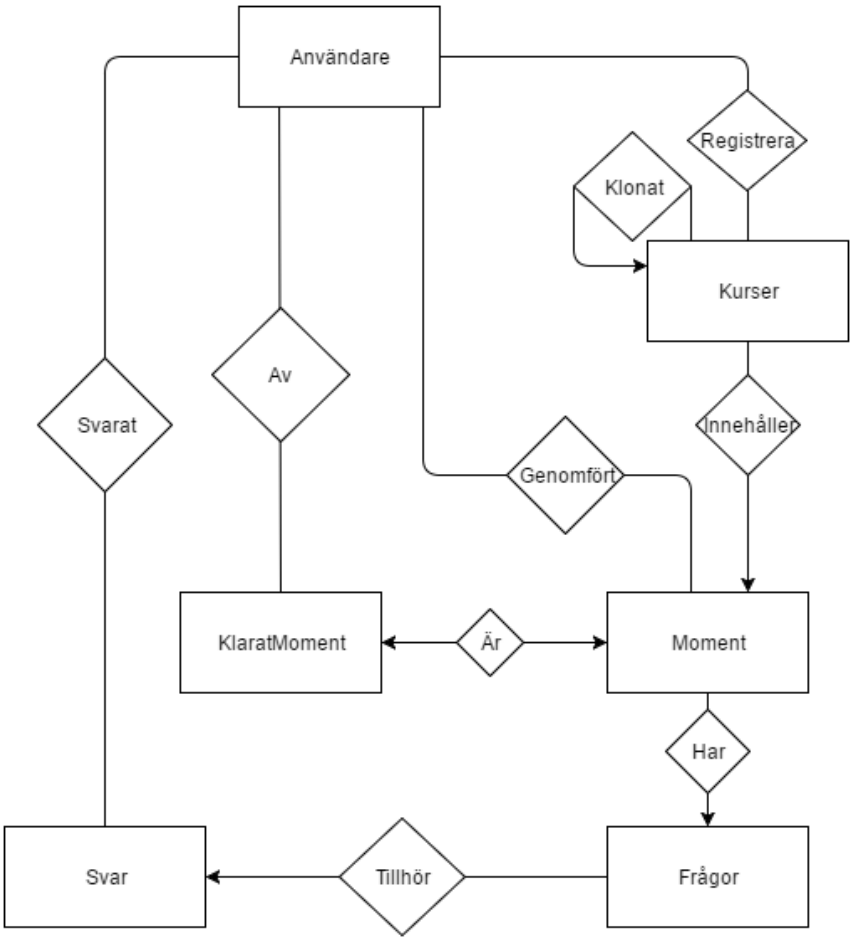
Utöver det här säkerhetstestet har flera kontrollertester skrivits till de resterande flikarna.Totalt har sex flikar skapat i REST API:n, var och en av dem testas med egna kontrollertester. Till skillnad från JUnit som används i säkerhetstestet, testas kontrollerna med integrationstest av Springs standardtestmetoder. En klass kallad för MockMvc fungerar som en basmotor i Springs MVC test och har som funktion att skapa nya integrationstester för kontroller.

6. Databas

Databasen som användas under arbetet är postgreSQL. Entiteter och attribut har överförts från en tidigare använd databas i Derby, vilket är ett standarddatabashanteringsverktyg i Netbeans. [13][3]

Under examenarbetet har ytterligare entiteter skapas baserat på den tidigare databasen. En av dem är “Answer” som nu har blivit en egen entitet. Entiteten innefattar information om bl.a. id, svar, rätt/fel, tillhörande fråga och svarande spelare. Vid varje spelomgång kommer svar från spelaren skickas och skrivas in i databasen m.h.a. REST API. Dessa kommer sedan presenteras som ett procentvärde i Like To Learns webbapplikation.

FinishedMoment är en annan entitet som har tillkommit under arbetes gång. Genom att lagra antalet klarade moment i den kan en användare fortsätta att träna där han/hon lämnade kursen sist.



Figur 6. En grundlig översikt av Like To Learns databas. Figuren förenklas i syfte att ge en lättfattlig uppfattning av databasens uppbyggande.

7. Resultat

7.1 Serverlösningen med ett REST API i Java

Som resultat har en fungerande serverlösning implementerats med ett REST API byggd med Spring som ramverk. Lösningen skyddas med ett enkelt säkerhetsförslag rekommenderat av Spring. [18] Lösningen fungerar på så sätt att Unityspelet skickar ut ett hårdkodat servernamn och lösenord till REST API:ns auktoritetsflik, dvs. /oauth/token i det fallet. REST API svarar i sin tur med en korrekt accesstoken som Unityspelet måste ha med i alla senare förfrågningar till andra kontrollerflikar.

Vidare har sex kontrollerflikar implementerats för att utföra den grundläggande dataöverföringen mellan Unitysetpel och databasen. Dessa kontrollerflikar har som uppgift att både skicka och ta emot data från databasen, ett exempel är funktionen som mottar statistik utskickad från Unityspelet för att spara undan den i databasen.

För att säkerhetsställa REST API:s funktionaliteter har både JUnit och Spring test tillförts som testramverk i denna serverlösning. I arbetet har JUnit använts främst inom enhetstester för säkerhet. Medan Spring test, dvs. Springs eget testramverk, används för att utföra integrationstester för olika kontrollerflikar. Resultat kan uppfattas som mycket positivt utifrån testresultatet. Båda testsorterna går genom som de ska.

7.2 Implementationen av en klientlösning i Unity

Att implementera en klientlösning i Unity med hjälp av C# har visat sig vara ett fungerande sätt att få ett Unityspel att kommunicera med ett REST API.

För att öka säkerheten frågar Unityklienten efter ett accesstoken m.h.a. ett krypterat servernamn och serverlösenord. Detta sparas sedan undan för att användas i alla de kommande anropen.

Anrop görs till REST API:n för att fråga efter nya frågor till spelet samt för att skicka användarnas svar.

För att testa att klientlösningen fungerar som den ska har tester gjort med testramverket NUnit.

7.3 Förbättringen av databasen

I arbetet har en befintlig databas utökas med två entiteter, nämligen svar och klaratmoment. I entitet svar lagras alla svar som spelare har svarat i Unityspel genom REST API. Relationen mellan svar och spelare lagras också för att databasen ska hålla koll på vilken spelare har svarat på detta svar. Kopplingen mellan svar och fråga registreras på samma sätt i databasen. Likaså gäller entiteten klaratmoment. Klaratmoment är kopplad till en existerande momentkod och den användaren som har klarat det momentet. På så sätt lagras all användares "sparande" i databasen. Användare kan då fortsätta på samma moment han/hon tränade sist genom att läsa in dessa klaratmomenten med hjälp av REST API.

7.4 Övrigt
Extra vyer för föräldrar samt presentationer av statistik gjordes genom att kod för att hämta resultat från databasen implementerades i den redan befintliga webbapplikationen.

Användarundersökningen visade att både lärare och föräldrar ser områden då Like To Learn kan bli till nytta, men även vissa risker. Den kan bli till nytta för lärare som vill få in statistik om hur mycket olika elever har förstått, samt för föräldrar som vill veta hur det går för deras barn i skolan. En risk kan till exempel vara om en elev skapar ett lärarkonto som andra elever vill ansluta sig till. Eleven kan då i detta lärarkonto skapa en mobbningskurs åt andra, vilket leder till risken av näthat.

8. Diskussion och slutsats

8.1 Hållbar utveckling
Like To Learn är en produkt som är skapad för att ge barn i 10-12:års åldern bättre förutsättningar för goda studieresultat. Tanken är att den ska distribueras gratis och därför ge barn ur olika samhällsklasser samma möjlighet att använda den. Produkten har möjlighet att översättas till många språk, varpå den kan ge barn från hela världen ett nytt bekvämt redskap under skolgången. Även barn som i dagsläget inte har tillgång till skola kan ta del av produkten. På så sätt kan produkten ses som etisk.

Miljömässigt är produkten gjord för att inte dra elektricitet i onödan. Programvaran är byggd för att vara stabil, dvs. inte krascha, så att datorer inte blir stående dragandes på el i onödan.

Om Like To Learn används istället för traditionella läxförhör på papper kan produkten bidra till minskad pappersanvändning. För att vara säker på att det är mer miljövänligt bör det kontrolleras hur användarnas elektroniska utrustning har tillverkats. Det är också mycket viktigt att se till att den utrustningen tas om hand på ett miljövänligt sätt den dag den inte längre går att använda. På grund av dessa två okända faktorer är det omöjligt att säga om bruket av Like To Learn i slutändan blir mer miljövänligt än klassiskt bruk av papper.

Helt säkert är dock att om Like To Learn används istället för traditionella pappersläxförhör i skolan, minskar det lärarnas tid för rättning. Det bidrar till en bättre arbetsmiljö för lärare.

8.2 Serverlösningen i Java
I arbetet med Java har Spring visat sig vara ett funktionellt ramverk med god struktur och bra information om hur man kommer igång. Det har visat sig mycket användbart när det gäller ett REST API som läser och skriver i en databas. Det finns gott om stöd och hjälp att komma igång. Att använda det ramverket tillsammans med Maven och Tomcat har också visat sig vara ett lyckat val.

Det märks att Springs utvecklare lagt ner mycket möda på designa så att det blir enklare att testa applikationerna som skapas. Även när det gäller test finns det god

information som gör det lätt att starta. Att testramverket JUnit tillfördes gjorde det lätt för Like To Learns utvecklare att välja det pga. tidigare positiva erfarenheter av just det testramverket.

Eftersom Spring var enkelt och behändigt att bygga upp ett REST API med, gjordes valet att även använda Spring Security. Detta var på samma sätt som övriga delar av ramverket lätt att ta till sig och börja använda.

Like To Learn innehåller nu ett hårdkodat användarnamn och lösenord för att öka säkerheten. Bedömningen är att Like To Learn inte är i behov av den allra högsta säkerheten, men ändå finns behovet av att se till att inte vem som helst kan hämta information hur som helst. Meningen är att bara de Unityklienter som har Like To Learns källkod ska skicka rätt lösenord och därmed få tillgång till REST API:n.

8.3 Klientlösningen i Unity

Att implementera en klientlösning i det befintliga Unity-spelet har fungerat väldigt bra Enkelheten att använda www-form gjorde det lätt att göra REST-anrop därifrån.

Det var också lätt att skapa kod för online/offline mode här eftersom det erbjöds enkla sätt att skriva kod som pingar olika adresser. Valet att pinga REST API:ns IP-adress gör att Unityspelet sparar undan all information även i de fall då kommunikationen avbryts p.g.a. att REST API:n för tillfället inte är uppe.

Svårigheten att hitta en lämplig parser kan inte ses som ett problem i Like To Learns fall, eftersom problemet avhjälptes mycket enkelt genom att utveckla en egen parser. Kanske kan det bli ett problem i en mer avancerad produkt.

8.4 Databasen

För att hantera alla användaruppgifter som kommer in från Unityspelet krävs det en stark databas, vilket inte fanns i de tidigare projekten. Därför beslutades det att övergå från Derby till PostgreSQL. Anledning till att just postgreSQL har valts är att den har en bättre prestanda för större datamängder och god hanteringsförmåga när det gäller avancerade SQL-förfrågningar. Detta har lett till att Like To Learn idag har en välfungerande databas.

8.5 Användaren

Alla de användare som deltagit användarundersökningen har uttryckt att de kan ha nytta av Like To Learn. Detta kan ses som en indikation på att Like To Learn kan vara en användarvänlig produkt. Olika önskemål har dock lyfts fram. Under arbetets gång har vissa av önskemålen tillgodosetts, som t.ex. lärarnas om att kunna se hur lång tid det tog för varje enskild elev att klara varje enskilt moment. Med hjälp av de metoder vi använt kan man utan problem tillgodose ett sådant önskemål.

På liknande sätt skulle man t.ex. kunna ta fram de utvecklingskurvor och statistik över tid som efterfrågades av föräldrar. På grund av tidsbrist har detta dock inte tagits fram under det här arbetets ram. Endast vissa funktionaliteter har valts ut för att visa på de valda teknikernas möjligheter.

Under användarstudien framkom även vissa risker med produkten. Alla lärarkonton som skapas är även administratörskonton för de elevkonton som väljer att ansluta sig till det. När man har skapat ett elevkonto kan man nämligen själv välja att ansluta sig till ett lärarkonto. Detta för att kunna svara på lärarens frågor, visa läraren sina resultat m.m. Att skapa ett sådant lärarkonto är extremt enkelt. Därför finns risken att en elev skapar ett lärarkonto för att sedan skapa egna "kurser" till andra elever. Om det finns andra elever som aktivt väljer att ansluta sig till en elevs lärarkonto, kan ett sådant konto användas till olämpligt bus.

Frågan är om detta är ett egentligt problem. Vill elever ansluta sig till ett lärarkonto som en klasskompis har skapat och rekommenderar? I användarstudien framkom dock rädsla för att detta skulle kunna användas för att uttrycka näthat. Det vill säga, att någon elev skulle skapa ett lärarkonto, starta en kurs, be kompisar ansluta sig och sedan lägga in frågor som t.ex. innehåller elaka skämt som klasskompisar eller liknande.

Ett enkelt sätt att minska risken för de oväntade administratörer som nämndes är att skapa två olika webbsidor, en där elever kan skapa sina inloggningsuppgifter, samt en annan där lärare kan skapa sina. Lärarnas internetadress bör vara lite krångligare att komma ihåg är elevernas, och behöver heller aldrig nämnas inför eleverna. Detta skulle göra att idén att skapa ett administrationskonto inte presenterar sig självt på samma sätt som det nu gör. Allra bäst vore kanske om lärarkonton bara kunde skapas med hjälp av godkända mailadresser. Detta kräver förstås att någon administratör lägger in vilka mailadresser som är godkända.

Eftersom det här arbetet inte innefattar förbättringar av webbapplikationen, har denna åtgärd för att ytterligare förbättra produkten Like To Learn ännu inte genomförts. Det är en kommande vidareutveckling.

8.6 Kritisk diskussion

Om detta arbete skulle göras igen, med samma kunskaper och syfte, skulle webbsidan ha tagits med i bearbetningen. Att utveckla olika delar av produkten samtidigt gör att man lättare kan anpassa dem efter varandra. Om arbetet istället hade varit upplagt på det sättet att både REST API:n och webbapplikationen utvecklades samtidigt, skulle problemet med de oönskade administratörerna kunnat minskas redan under det här arbetets gång. Nu skjuts det istället på framtiden.

En fördel med att inte blanda in webbapplikationen i det här arbetet har varit att REST API kunnat utvecklas och testas ordentligt så dess buggar har eliminerats, detta på bekostnad av att webbapplikationens buggar fortfarande finns kvar.

Om djupintervjuer hade använts istället för enkätundersökningen, hade troligtvis mer kunskap om Like To Learns användnings- och utvecklingsmöjligheter kunnat sammanställas. Mer användbar information har framkommit genom samtal än genom enkäten. Dessa kan dock inte redovisas i det här arbetet på grund av bristande dokumentation.

För övrigt har arbetet fungerat väl. Valet av Spring, Tomcat och Maven tillsammans har visat sig vara väldigt lyckat.

8.7 Slutsatser

Under arbetets gång har en klient- och serverlösning i Unity och Java utvecklats. Produkten har testats av flera olika typer av användare, så som elever, lärare och föräldrar.

Problemen som stöttes på under arbetsprocessen har varit minimala, mycket tack vare ramverket Spring. Det har visat sig vara enkelt att arbeta med samt mycket funktionellt. Med hjälp av Springs befintliga exempelprojekt kommer man snabbt i gång med utvecklingen av ett REST API. Under utvecklingen gjordes även en förenklad säkerhetslösning med Springs rekommenderade förslag. Ramverket Spring fungerar mycket väl när man konstruerar ett REST API som ska läsa och skriva i en databas.

Designmönstret i Spring underlättar även vid skapandet av enhetstester i ett sådant REST API. För att skriva säkerhetstest tillförs testramverket JUnit som är väl anpassat för denna uppgift. Slutsatsen är att det kan uppfattas som mycket positivt. Tillsammans med Springs egna testmetoder skapar JUnit en omfattande testram kring säkerhet och kontroller.

Även när det gäller klientlösningen i Unity måste slutsatsen bli att verktyget fungerar väl. Det går lätt kan göra specifika anrop till ett REST API.

Klienter utvecklade i Java fungerar väl tillsammans med en klientlösning implementerad i Unity i en produkts så som till exempel Like To Learn. Att kombinera en backend skriven i Java med en Unityklient skriven i C# går mycket bra. Alla de användare som deltagit i vår studie har uttryckt att de på något sätt kan ha nytta av en sådan produkt.

9. Framtida utveckling

Efter examensarbetet planerar vi att släppa Like To Learns spel på två plattformar, nämligen PC och Android. Webbapplikationen kommer flyttas till en ny domän. På startsidan syns nu en kort beskrivning av själva spelet. Under den kommer en länk till nedladdning av Unityspelet stå, tillsammans med en mailadress som mottar feedback från användarna kontinuerligt. Tanken är att deltagarna från användarundersökning ska få möjlighet att testa spelet och fortsätta skicka kommentarer om vad som kan bli förbättrat.

Utvecklingsmöjligheterna av Like To Learn är mycket stora. I webapplikationen kan man till exempel skapa ny statistik beroende på vad användarna önskar. Bland föräldrarna som var med i användarstudien framkom önskemål om att en utvecklingskurva kan vara något som är intressant att se. Lärarna föreslog att elevlistorna sammanställs tydligare klassvis. I enkätundersökning samlas flera sådana goda råd som skulle kunna bli möjliga utvecklingsplaner framöver.

Även vissa risker som beskrivs i användarenkäten, som tillexempel risken att elever skapar lärarkonton, skulle kunna minskas genom vissa åtgärder, t.ex. skapa en särskild hemsida för lärare. Man skulle också kunna införa att lärarkonton bara kan skapas med godkända mailadresser som någon administratör har lagt in. Detta är någonting som ännu inte har hunnits med och blir därför en sak som ska göras i framtiden.

Förutom dessa tilläggningsfunktioner kan funktionaliteter för nya typer av användare läggas till, som t.ex. för rektorn. Detta skulle leda till en ökad målgrupp för produkten och på så sätt skapas ännu större möjligheter för utveckling.

I backend kan REST API vidareutvecklas med en högre säkerhetsnivå än den nuvarande. De hårdkodade inloggningsuppgifterna skulle kunna vara saltade för att öka säkerheten. En risk som alltid finns är att databasen i fråga skulle kunna bli intressant att hacka. Kanske användarna använder sig av samma användarnamn och lösenord i Like To Learn som på andra ställen. I så fall skulle en hackare kunna ha nytta av dem för att komma åt deras konton på andra ställen. Inom det här arbetet har valet gjorts att inte lägga allt för mycket tid på säkerhetstänkande. Istället har fokus legat på att bygga upp grundläggande funktioner.

9.1 Framtida användare

Under arbetets gång har även en kontakt tagits med en Tv-kanal som sänder skolprogram avsedda för syriska flyktingläger i Mellanöstern. De har delgivits rättigheter att använda Like To Learn utan kostnad. Genom Like To Learn skulle lärarna som undervisar genom Tv:n kunna få tillbaka statistik från sina elever. På så sätt skulle de kunna få uppfattning om vad som behöver repeteras samt om vad som har gått hem hos tittarna.

Ett problem är dock internetuppkopplingarna i flyktinglägren. Det finns fungerade satelituppkopplingar och Tv-sändningar, men internet är inte lika pålitligt och lättåtkomligt. Frågan kom därför upp om produkten kanske är mer lämplig för att

stödja andra grupper av potentiella tittare hos dem. Exempelvis flickor som inte går i skolan med befinner sig regioner i närheten av lägren. Om de är på platser där internetuppkopplingarna är stabila skulle de kunna ha nytta av Like To Learn i framtiden.

10. Referenser

[1] Azure.microsoft.com. (2016). *Vad är Azure – den bästa molntjänsten från Microsoft | Microsoft Azure*. [online] Available at: https://azure.microsoft.com/sv-se/overview/what-is-azure/ [Accessed 29 April 2016].

[2] *C# Language Specification* (PDF) (4th ed.). Ecma International. June 2006. Retrieved April 14, 2016.

[3] Db.apache.org. (2016). *Apache Derby*. [online] Available at: https://db.apache.org/derby/ [Accessed 13 May 2016].

[4] Forsberg, J., Johansson. F., Leiulfsrud. L., Luong. Q., & Sjöberg. J. (2016) *Datorspel i Unity*. Göteborg: Chalmers University of Technology, Datateknik (högskoleingenjör) (Ett projektarbete på Dataingenjörsprogrammet)

[5] Forsberg, J., Johansson. F., Leiulfsrud. L., & Luong. Q. (2016) *Projekt webbapplikation*. Goteborg: Chalmers University of Technology, Datateknik (högskoleingenjör) (Ett projektarbete på Dataingenjörsprogrammet)

[6] Google.com. (2016). *Google Drive - Cloud Storage & File Backup for Photos, Docs & More*. [online] Available at: https://www.google.com/drive/ [Accessed 22 April 2016].

[7] Guides.github.com. (2016). *Understanding the GitHub Flow · GitHub Guides* . [online] Available at: https://guides.github.com/introduction/flow/ [Accessed 22 April 2016].

[8] Java.com. (2016). *What is Java and why do I need it?*. [online] Available at: https://java.com/en/download/faq/whatis_java.xml [Accessed 29 April 2016].

[9] Johnson, R. (2005). "Introduction to the Spring Framework". [online] Available at: http://www.theserverside.com/tt/articles/article.tss?l=SpringFramework/ [Accessed 14 April 2016].

[10] Junit.org. (2016). *JUnit - About*. [online] Available at: http://junit.org/junit4/ [Accessed 18 May 2016].

[11] Maven.apache.org. (2016). *Maven – Introduction*. [online] Available at: http://maven.apache.org/what-is-maven.html [Accessed 13 May 2016].

[12] Msdn.microsoft.com. (2016). *Structured Query Language (SQL)*. [online] Available at: https://msdn.microsoft.com/en-gb/library/windows/desktop/ms714670(v=vs.85).aspx [Accessed 29 April 2016].

[13] Netbeans.org. (2016). *An Introduction to NetBeans*. [online] Available at: https://netbeans.org/about/ [Accessed 13 May 2016].

[14] Nunit.org. (2016). *NUnit - Home*. [online] Available at: http://www.nunit.org/ [Accessed 18 May 2016].

[15] Orrje, J., & Persson, D. (2007). Java , ramverk och komplex webutveckling En helhetsstudie av utvecklingsprocessen. Uppsala : Uppsala universitet

[16] Postgresql.org. (2016). *PostgreSQL: About*. [online] Available at: http://www.postgresql.org/about/ [Accessed 29 April 2016].

[17] Project, A. (2016). *Apache Tomcat® - Welcome!*. [online] Tomcat.apache.org. Available at: http://tomcat.apache.org/ [Accessed 13 May 2016].

[18] Spring.io. (2016). *spring.io*. [online] Available at: https://spring.io/ [Accessed 3 Jun. 2016].

[19] Tutorialspoint.com (2016). *Eclipse Overview*. [online] Available at: http://www.tutorialspoint.com/eclipse/eclipse_overview.htm [Accessed 29 April 2016].

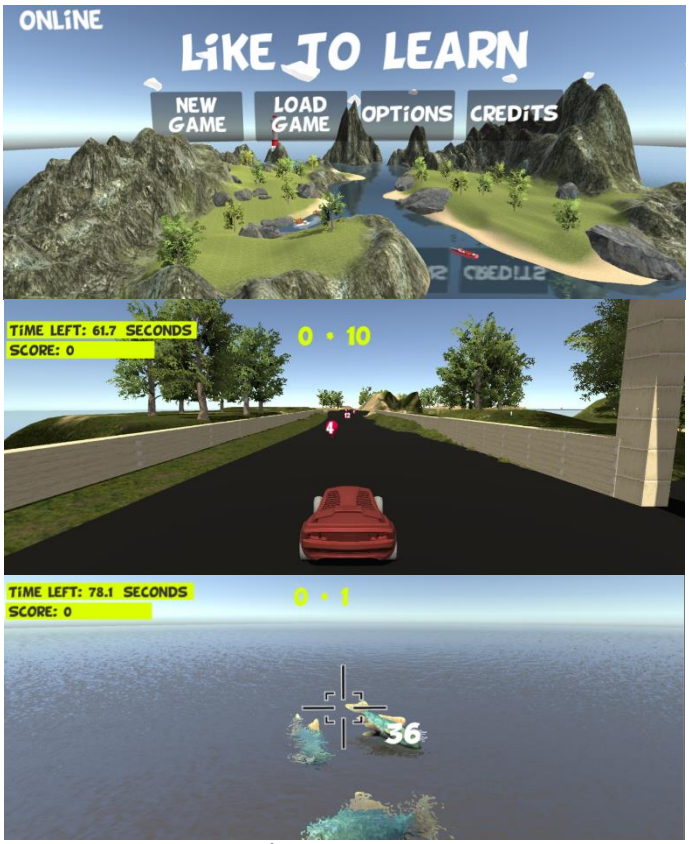
[20] Ubuntu.com. (2016). *About Ubuntu | Ubuntu*. [online] Available at: http://www.ubuntu.com/about/about-ubuntu [Accessed 13 May 2016].

[21] Unity3d.com. (2016). *Unity - Company*. [online] Available at: https://unity3d.com/company [Accessed 14 April 2016].

[22] Visualstudio.com. (2016). *Visual Studio - Microsoft Developer Tools*. [online] Available at: https://www.visualstudio.com/ [Accessed 14 April 2016].

11. Bilagor

11.1. Bilaga 1: Bilder från Unityspelet



Bilder från LIke To Learns spel.
Här exemplifieras spelets frågor mha nollans gångertabell.

11.2. Bilaga 2: Undersökningsenkäten

Följande frågor gavs till olika användare under användarstudien:

Två lärare och två föräldrar svarade på enkäten.

1. Vilken typ av användare är du när du använder Like To Learn?

Elev Förälder Lärare

2. Vad skulle du kunna använda spelet till?

3. Vad skulle du vilja kunna reglera i spelet?

4. Vilken sorts statistik skulle du vilja se kunna se från spelet?

5. Skulle spelet kunna förenkla någonting för dig? I så fall vad?

6. Ser du några risker med att använda spelet? I så fall vilka?

11.3. Bilaga 3: Lärarundersökning

Svar från lärare som deltog i användarstudien:

Fråga	Lärare 1	Lärare 2
Vad skulle du kunna använda spelet till?	Jag skulle vilja använda Like To Learn för individuella läxor till mina elever. Spelet ger möjlighet för olika elever att öva olika saker olika. Utan mer arbete för mig ser spelet till att rätt elev övar på rätt nivå.	Jag skulle kunna använda det för svaga elever som behöver öva mer i sin egen takt.
Vad skulle du vilja kunna reglera i spelet?	Jag vill kunna bestämma att en viss läxa ska "köras" igen om jag bedömmer att en elev behöver öva mer på ett moment han/hon gjort en gång för länge sedan.	Jag skulle vilja kunna bestämma lägsta godkända nivån för ett moment, ska det verkligen vara 100 % jämt? Kanske vore det bra om man kunde ha många rätta svar på en och samma fråga ibland. Det skulle vara bra om läraren kunde bestämma vilken del av Unityspelet som ska spelas just nu.
Vilken sorts statistik skulle du vilja kunna se från spelet?	Jag vill se hur det går för alla mina elever i varje enskilt moment och fråga.	Det förenklar för mig om elevernas statistik redovisas klassvis. Jag skulle gärna vilja se statistiken redovisad läxförhör för läxförhör, samt fråga för fråga. Det vore också bra att kunna se vilka klasser jag kört en viss kurs med. Som lärare vill jag veta hur många procent som klarat av lägsta nivån, samt vilka det var som inte klarade den. Jag vill även kunna gå in och kolla varje elevs resultat.

		Det är intressant för mig att veta om många elever ger samma felsvar, eller om många svarar fel på samma fråga. Om en elev alltid ger samma felsvar, vore det också intressant att veta. Att se hur lång tid det tog för varje enskild elev att klara ett visst moment, skulle ge en indikation på hur lätt/svårt det var.
Skulle spelet kunna förenkla någonting för dig? I så fall vad?	Det skulle minska den tid jag lägger på rättning och snabbare ge mig information om vad jag behöver hjälpa min elever med.	Spelet ger mig en tydlig översikt över mina elevers resultat. Jag kan hålla mig uppdaterad på elevers resultat i realtid utan att behöva springa runt så mycket.
Ser du några risker med att använda spelet? I så fall vilka?	Att elever svarar fel i spelet kan ibland bero på svårigheter i spelet och inte på att de inte kan frågan. Om läraren inte är medveten om detta kan spelstatistiken ge missvisande information.	Om elever loggar in med fel namn eller andra användares namn, kan det vara riskabelt om den eleven redigerar eller ändrar uppgifter för andra användare. Det kan bli näthat om en elev skapar ett lärarkonto och mobbningskurser till andra elever.

11.4. Bilaga 4: Föräldrarundersökning

Svar från föräldrar som deltog i användarstudien:

Fråga	Förälder 1	Förälder 2
Vad skulle du kunna använda spelet till?	Som förälder skulle jag kunna använda spelet till att följa hur det går för min dotter när det gäller läxan. Jag kan se vad hon har lätt för respektive svårt för och på så sätt stötta när det behövs.	Framför allt skulle jag vilja se mitt barns utveckling över tid. Jag skulle också vilja kunna se hur han/hon ligger till i förhållande till övriga i klassen. (Kanske inte är pedagogiskt försvarbart men som förälder hade det varit önskvärt) Jag skulle också vilja se om han/hon har gjort de kurser (exempelvis läsläxor) som är tänkt för veckan.
Vad skulle du vilja kunna reglera i spelet?	Oj, svår fråga. Bygga en föräldrar vy som t.ex. innehåller feedback från läraren via systemet tillsammans med statistiken.	Ser inte något behov att reglera som förälder.
Vilken sorts statistik skulle du vilja kunna se från spelet?	- Jag vill se statistik kring respektive ”kurs” som hon gör, resultat och tiden hon lägger. - Hur frekvent hon spelar spelet, dvs pluggar. - Och statistik över tid, hur ser utvecklingskurvan ut. - Feedback från läraren kring hur mitt barn presterar.	Status på utveckling, Utläsa trender (t.ex. svårt för 7:ans gånger tabell)
Skulle spelet kunna förenkla någonting för dig? I så fall vad?	Det skulle göra det roligare för dottern att plugga t.ex. glosor eller multiplikation, och på så sätt förenkla för mig som förälder som då inte behöver förhöra. Det skulle också ge mig som förälder en kontroll att läxan blir gjord och	Spelet skulle ge mig en tidig fingervisning om att det finns saker att fokusera på. Jag ”slipper” ta kontakt och fråga.

	inlämnad till läraren, idag vet jag inte om stencilen med mattetal kommer fram till läraren.	
Ser du några risker med att använda spelet? I så fall vilka?	Risken är att eleven blir sittandes länge framför ipaden, barnen fastnar ännu mer vid paddan.	Jag kan inte se några risker.