YRKESHÖGSKOLAN NOVIA

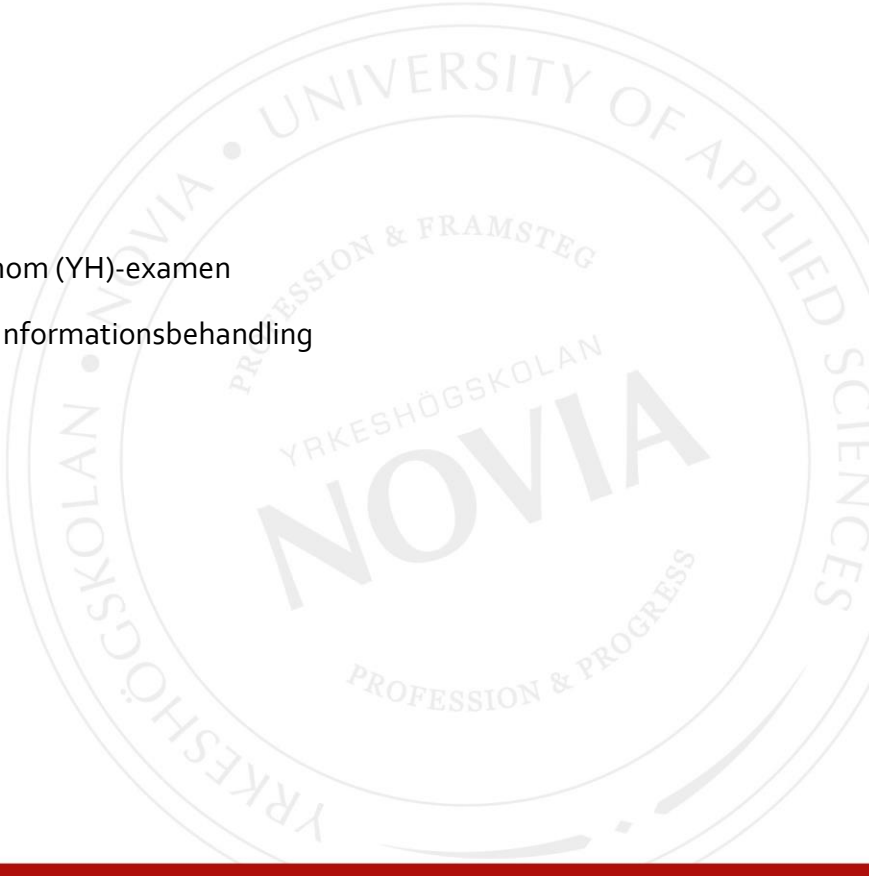
Webbaserat interaktivt inlärningsspel baserad på öppen webbteknologi i HTML5

Daniel Kouvo

Examensarbete för Tradenom (YH)-examen

Utbildningsprogrammet i Informationsbehandling

Raseborg 2018



EXAMENSARBETE

Författare: Daniel Kouvo

Utbildning och ort: Informationsbehandling, Raseborg

Handledare: Klaus Hansen

Titel: Webbaserat interaktivt inlärningsspel baserad på öppen webbteknologi i HTML5

Datum 17.05.2018 Sidantal 34

Abstrakt

Examensarbetet handlar om utvecklandet av ett webbaserat interaktivt inlärningsspel som bygger på öppen webbteknologi i HTML5. Spelet är ett inlärningsspel vars målgrupp är skolelever. Spelet handlar om köttindustrin och köttproduktionens olika skeden. Spelet fungerar som ett komplement till läromaterial. Syftet är att på ett intressant och roligt sätt få eleverna att ta till sej och bekanta sej med informationen.

Bakom spelet finns en Wordpress baserad backend där det är enkelt att hantera de frågor, svar och innehåll som presenteras i spelet. Wordpress fungerar samtidigt som materialbank med lösenordskyddade undersidor där lärare kan ladda ner material i bl.a. PDF format.

Spelet körs tekniskt sett i Wordpress miljö men fungerar självständigt ovanpå Wordpress. Frågor, svar och innehåll hämtas till spelet från WordPress. Spelmiljön körs i webbläsare i en HTML5 canvas som tack vare den underliggande koden uppdateras i realtid och skalas responsivt. Spelet renderas till en HTML5 canvas via Javascript. Det modulära CreateJS paketet har använts för att möjliggöra animation, interaktivitet och responsiv design för HTML5 canvas.

Projektet upprätthålls i en Git repository. För projektets lokala utvecklingsmiljö används den portabla virtuella utvecklingsmiljön Vagrant. Projektets uppdateringar publiceras automatiskt från sin Git repository till en dedikerad Wordpress servermiljö med hjälp av den automatiserade Jenkins servertjänsten.

Examensarbetet beskriver förverkligandet av projektet från start till slut samt tekniken och utvecklingsmiljön.

Språk: Svenska Nyckelord: CreateJS, Javascript, HTML5, Wordpress, Spel

---

BACHELOR'S THESIS

Author: Daniel Kouvo

Degree Programme: Business Information Technology, Raseborg

Supervisor(s): Klaus Hansen

Title: Web-based Interactive E-learning Game Based on Open Web Technologies in HTML5

Date 17.05.2018 Number of pages 34

Abstract

This thesis covers the development of an interactive web-based educational game that is based on open web technologies in HTML5. The target group for the game is students. The game informs students about the meat industry and the different stages of meat production. The game is a complement to traditional educational materials. The aim is to provide students with an interesting and fun way of learning.

The underlying Wordpress backend provides an easy way to update or add questions, answers and content used in the game. Teachers can also download teaching materials in PDF format from password protected pages provided by the Wordpress CMS.

Technically the game runs from a Wordpress environment but works independently on top of Wordpress. Questions, answers and content used in the game are served from Wordpress. The game is rendered to a HTML5 canvas which gets updated in real time and has responsive scaling. This is done using Javascript and the CreateJS suite, which enables animation, interactivity and responsive design for the HTML5 canvas.

The project is maintained in a Git repository. Vagrant, which is a portable virtual software development environment, is used for local development. All updates are automatically deployed from the Git repository to a dedicated Wordpress hosting using a Jenkins automation server.

This thesis covers the development phases from start to end and the technology and development environment used in the creation of the project.

Language: Swedish Key words: CreateJS, Javascript, HTML5, Wordpress, Game

---

Innehållsförteckning

Figurförteckning

1 Inledning

Examensarbetet handlar om utvecklandet av ett beställningsarbete på ett webbaserat interaktivt inlärningsspel. Inlärningsspelet har jag utvecklat på en digibyrå där jag arbetat som webbutvecklare för projektet. Själva materialet och grafiska designen av spelet levererades av beställaren som ett Word dokument och en Adobe Illustrator fil. Utgående från det materialet har jag utvecklat projektet till ett interaktivt webbaserat inlärningsspel. Plattformen för inlärningsspelet baserar sej på en kombination av Wordpress, REST API, CreateJS och jsPDF.

1.1 Syfte

Examensarbetets syfte är att beskriva utvecklingsprocessen av inlärningsspelet och hur tekniken bakom spelet fungerar. Inlärningsspelet fungerar som ett komplement till läromaterial. I spelet ska elever på ett intressant och roligt sätt kunna bekanta sej med information om köttindustrin i Finland och köttproduktionens olika skeden.

2 Tekniken

Spelet körs tekniskt sett i Wordpress miljö men fungerar självständigt ovanpå Wordpress. Frågor, svar och innehåll hämtas till spelet från Wordpress via ett REST-gränssnitt. Spelmiljön körs i webbläsare i en HTML5 canvas som tack vare den underliggande koden uppdateras i realtid och skalas responsivt. Spelet renderas till en HTML5 canvas via Javascript. Det modulära CreateJS paketet har använts för att möjliggöra animation, interaktivitet och responsiv design för HTML5 canvas. Projektet upprätthålls i en Git repository. För projektets lokala utvecklingsmiljö används den portabla virtuella utvecklingsmiljön Vagrant. Projektets uppdateringar publiceras automatiskt från sin Git repository till en dedikerad Wordpress servermiljö med hjälp av den automatiserade Jenkins servertjänsten. I det här kapitlet beskrivs dessa tekniker och verktyg.

2.1 WordPress

WordPress är ett avancerat innehållshanteringssystem (CMS) som i grunden baserar sej på PHP. Wordpress är en fri programvara med öppen källkod. WordPress hör till de populäraste innehållshanteringssystemen. Tusentals webbplatser och bloggar är baserade på WordPress. Enligt wordpress.org (2018) består 30% av webben av WordPress. WordPress designades

ursprungligen som en plattform för bloggar men har under åren utvecklats till ett avancerat innehållshanteringssystem. WordPress är väldigt flexibelt och erbjuder nästan obegränsade möjligheter till uppbyggnad av olika slags projekt. Själva inlärningsspelet fungerar i huvudsak tekniskt rätt fristående från WordPress men frågedatabasen, materialbank, SEO optimering, sidhantering m.m. gör inlärningsspelet till ett komplett webbprojekt.

2.1.1 REST API

REST API (Representational State Transfer Application Programming Interface) som numera finns integrerat i WordPress gör det möjligt att integrera WordPress i princip med vilken som helst applikation oavsett programmeringsspråk. Med hjälp av REST API kan t.ex. en extern applikation kommunicera med WordPress och bl.a. nå data från WordPress som JSON objekt. Inlärningsspelet utnyttjade i prototypskedet denna möjlighet för att hämta de frågor, svar och innehåll som finns inmatade i WordPress backend.

2.1.2 Advanced Custom Fields PRO

ACF (Advanced Custom Fields) PRO är ett flexibelt tillägg (plugin) till WordPress som möjliggör skapandet av skräddarsydda fält med avancerad funktionalitet. Med hjälp av tillägget kan man utöka möjligheterna i WordPress och t.ex. bygga skräddarsydda verktyg i backend. Med hjälp av ACF har jag byggt upp bl.a. en frågedatabas i WordPress backend vilket spelet utnyttjar. Bång (2017) påpekar att man med ACF kan göra sitt tema dynamiskt.

2.1.3 Custom Post Type UI

WordPress erbjuder två olika inläggstyper som standard, sidor (Pages) och inlägg (Posts). Detta kan utökas genom att skapa skräddarsydda inläggstyper (Custom Post Types). Med hjälp av CPT UI (Custom Post Type UI) kan detta göras direkt i WordPress backend. Det är även möjligt att redigera och hantera de olika inläggstyperna, samt bestämma vilka funktioner de olika inläggstyperna stöder. Custom Post Type UI gör det även möjligt att koppla på REST API integrationen till en skräddarsydd inläggstyp. Frågedatabasen i inlärningsspelet är uppbyggd i en custom post type. Bång (2017) påpekar att CPT UI tillsammans med ACF är en kraftfull kombination.

2.1.4 All 404 Redirect to Homepage

All 404 Redirect to Homepage är ett WordPress tillägg som skapar en redirect till första sidan i WordPress för alla 404 träffar. Tillägget samlar även statistik på all trafik den omdirigerat och skapar en grafisk tidslinje över detta. Eftersom man vill att användarna skall hitta till spelet är det en bra idé att omdirigera eventuella felskrivningar till första sidan av WordPress där spelet startar. Statistiken ger en bra bild över hur vanligt och hur ofta felskrivningar skett samt vilka slags felskrivningar det varit fråga om.

2.2 CreateJS

CreateJS är ett paket med olika typer av modulära JavaScript-bibliotek som innehåller funktioner för att möjliggöra animation, effekter, användning av ljud m.m. i webbsidor. Paketet bygger på öppen webbteknologi, stöder alla moderna och många äldre webbläsare och är väldigt enkelt att ta i bruk. CreateJS hör till de viktigaste byggnadsstenarna i inlärningsspelet.

2.2.1 EaselJS

EaselJS är en del av CreateJS paketet. Det är ett JavaScript bibliotek som gör det möjligt att ta full nytta av Canvas elementet i HTML5 och gör det bl.a. enklare att arbeta med. Användbart för att bl.a. skapa spel, animationer, konst och andra grafiska upplevelser. EaselJS används i inlärningsspelet för att skapa spelets grafik och funktionalitet i ett HTML5 Canvas element.

2.2.2 TweenJS

TweenJS är en del av CreateJS paketet. Det är ett lättanvänt och kraftfullt JavaScript-bibliotek för att skapa effekter och animationer i HTML5. Kan användas fristående eller integrerat med EaselJS. I inlärningsspelet är alla effekter och animationer uppbyggda med hjälp av TweenJS.

2.2.3 SoundJS

SoundJS är ett JavaScript-bibliotek som erbjuder ett lättanvänt applikationsprogrammerings-gränssnitt (API) med kraftfulla funktioner för att hantera och t.o.m. skapa ljud i webbprojekt. Kan även kombineras med PreloadJS för att hantera hämtningen av ljudfilerna. SoundJS är

en del av CreateJS paketet och stöd för dess eventuella framtida användning i inlärningsspelet har byggts in i spelet.

2.2.4 PreloadJS

PreloadJS är en del av CreateJS paketet. Det är ett JavaScript-bibliotek som hanterar hämtning av filer. PreloadJS ansvarar för att ladda in filer som t.ex. bild och ljud till ett webbprojekt och kan utnyttjas för att bl.a. bygga en s.k. loader. I inlärningsspelet ser PreloadJS till att ladda in alla nödvändiga filer innan själva spelet startar.

2.3 jsPDF

jsPDF är ett JavaScript-bibliotek som kan dynamiskt generera PDF filer. I slutskedet av inlärningsspelet tilldelas spelaren ett diplom baserat på vilken nivå i spelet spelaren lyckats uppnå. Det diplomet genereras dynamiskt och med hjälp av jsPDF kan samma diplom även generas som en PDF.

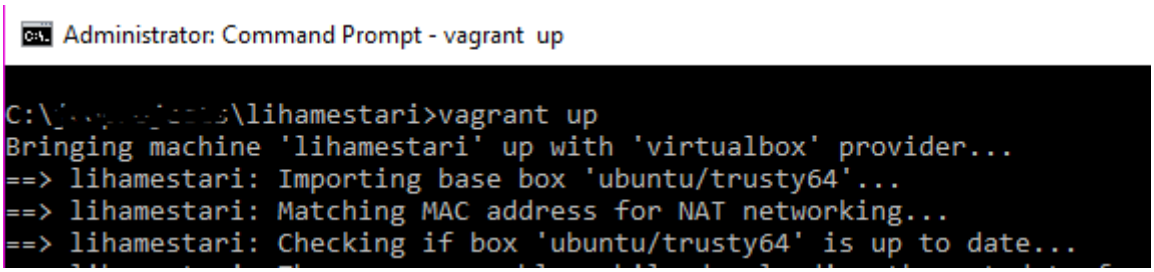
3 Utvecklingsmiljö

Detta kapitel beskriver projektets utvecklingsmiljö. Inlärningsspelet har jag utvecklat i en virtuell portabel utvecklingsmiljö, Vagrant i samverkan med Virtualbox. Den portabla utvecklingsmiljön möjliggör bl.a. att den kan upprätthållas i ett versionshanteringssystem och köras i flera olika lokala miljöer. Själva projektet kan upprätthållas i en skild repository i ett versionshanteringssystem vilket innebär att de inte är beroende av varandra men som en helhet utgör de en miljö där projektet körs lokalt i den virtuella utvecklingsmiljön. Den virtuella portabla utvecklingsmiljön är heller inte bunden till något visst operativsystem. Detta gör att det är enkelt att fortsätta utveckla projektet när som helst från i stort sett vilken som helst dator. Detta möjliggör även att flera personer vid behov kan utveckla på samma projekt i egna lokala miljöer. Eftersom själva projektet inte är bundet med den virtuella utvecklingsmiljön innebär det också att den kan vid behov bytas ut till något annat.

3.1 Vagrant

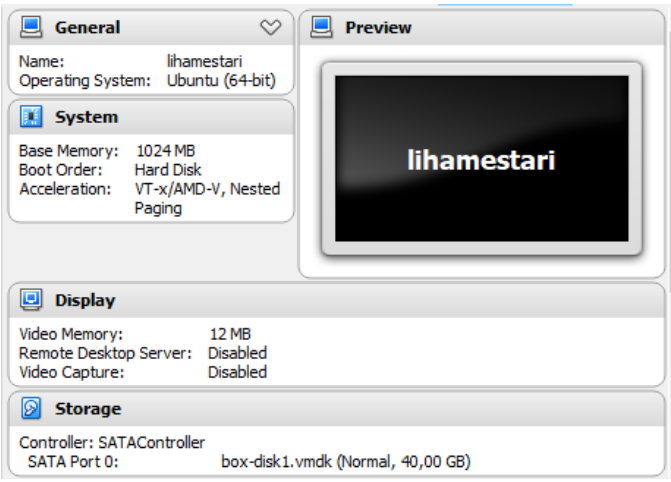
Vagrant är en mjukvaruprodukt med öppen källkod. Vagrant gör det både enkelt och effektivt att bygga och upprätthålla virtuella utvecklingsmiljöer. Vagrant används tillsammans med mjukvara utvecklad för att köra virtuella maskiner. Sådan mjukvara kan vara t.ex. VirtualBox, Hyper-V, Docker containers, VMware och AWS. I detta projekt har

jag använt mej av kombinationen Vagrant tillsammans med Virtualbox. Efter att grundkonfigurationen i Vagrant är gjord är det väldigt enkelt att start upp och stänga ner den virtuella utvecklingsmiljön. Vagrant startas genom att ange kommandot ”vagrant up”. Första gången detta körs byggs först den virtuella utvecklingsmiljön upp och därefter startar den upp. Den första starten tar därför lite längre tid då den t.ex. bygger upp en virtuell Linux servermiljö. Därefter kommer detta kommando helt enkelt att bara starta upp den virtuella utvecklingsmiljön vilket också innebär att den startar upp väldigt fort. Med kommandot ”vagrant halt” stoppar man den virtuella utvecklingsmiljön. Vagrant är ett mycket stabilt och pålitligt system men i vissa fall kan det hända att det uppstår något slags fel i den virtuella utvecklingsmiljön som orsakar en situation där den virtuella utvecklingsmiljön inte vill starta. Detta kan hända om man t.ex. stänger ner datorn men glömt att stänga ner Vagrant med kommandot ”vagrant halt”. Ifall det visar sej att den virtuella utvecklingsmiljön fått något fel är det enkelt att korrigera det genom att köra kommandot ”vagrant destroy” som raderar den virtuella utvecklingsmiljön som byggts upp. Därefter kan man åter igen köra kommandot ”vagrant up” (Se figur 1) vilket då återuppbygger den virtuella utvecklingsmiljön. Eftersom själva projektet och den virtuella utvecklingsmiljön inte är beroende av varandra raderas inget från projektet då ”vagrant destroy” körs. nshalv.com (2014) lyfter fram Vagrant som en toppen miljö för WordPress utveckling och testning.



Figur 1. Vagrant startas upp med kommandot vagrant up.

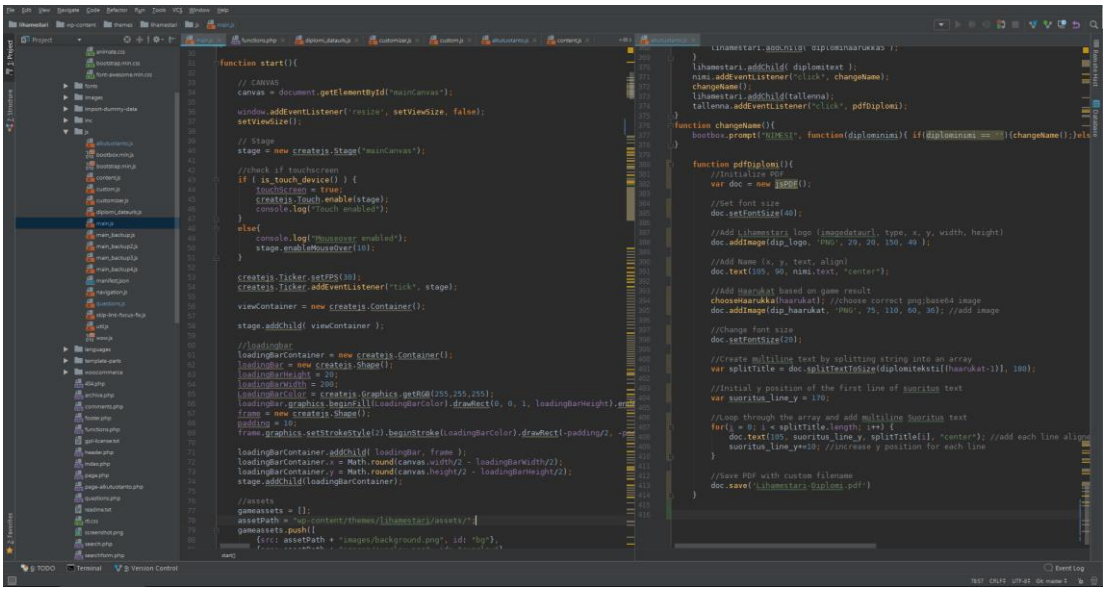
Virtualbox (Se figur 2) är en kraftfull x86 och Intel64/AMD64 virtualiseringsmjukvara. Virtualbox är den enda mjukvaran inom detta område med öppen källkod under GPL (GNU General Public License) och som finns fritt tillgängligt för allmänheten. Virtualbox kan köras bl.a. i Windows, Macintosh och Linux och kan köra massor av olika operativsystem virtuellt. I detta projekt har jag använt Vagrant som en portabel virtuell utvecklingsmiljö som i sin tur använder Virtualbox för att köra en Ubuntu server. Den virtuella Ubuntu servern fungerar som projektets webbserver i den lokala miljön.



Figur 2. Den virtuella maskinen i Oracle VM VirtualBox manager.

3.2 JetBrains PhpStorm

PhpStorm är en IDE (Integrated Development Environment) ämnad för webb-programmering med fokus på PHP. Trots fokuseringen på PHP och namnet är PhpStorm ett fullständigt verktyg även för webbprogrammering som inte involverar PHP. Eftersom jag jobbar rätt mycket med PHP och WordPress baserade projekt där PHP är en stor del av arbetet har PhpStorm blivit en naturlig del i mitt arbete. Därför valde jag att använda det även i detta projekt (Se figur 3). PhpStorm innehåller en hel del effektiva, kraftfulla och smarta funktioner som snabbar upp och underlättar olika skeden i programmering. PhpStorm har också bl.a. ett väldigt bra integrerat stöd för versionshantering, terminalintegration och olika slags pre-compilers.

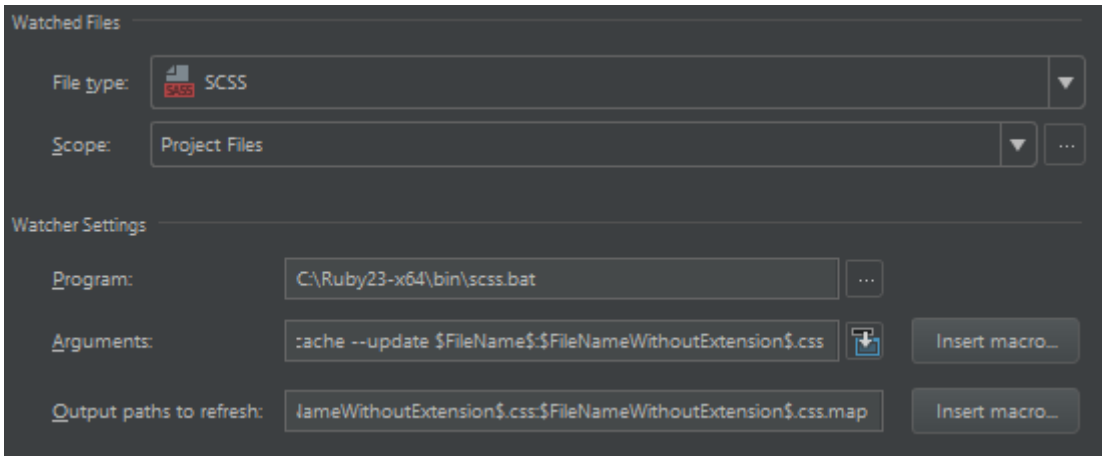


Figur 3. Inlärningsspelet under arbete i PhpStorm.

3.3 SCSS och Ruby

Ruby är ett objektorienterat programmeringsspråk. SCSS (Sassy CSS) är en del av SASS (Syntactically awesome style sheets) som är ett stilmallsspråk. Enligt Long (2011) bygger SCSS på samma syntax som CSS men innehåller många praktiska funktioner som t.ex. användning av variabler. Detta effektiverar arbetet och bidrar till att minska eventuella mänskliga misstag.

En webbläsare förstår sej inte på SCSS och därför måste SCSS kompileras till CSS. Från inlärningsspelets SCSS filer genereras en slutlig CSS fil med alla stilar. Detta generas med hjälp av Ruby som finns installerad i den lokala miljön och konfigurerad som en SCSS File Watcher i PhpStorm (Se figur 4). Då det görs ändringar i SCSS filerna i PhpStorm reagerar SCSS File Watcher konfigurationen på dessa. Med hjälp av Ruby kompilerar den en slutlig CSS fil. Dessutom varnar SCSS File Watchern om eventuella felskrivningar.



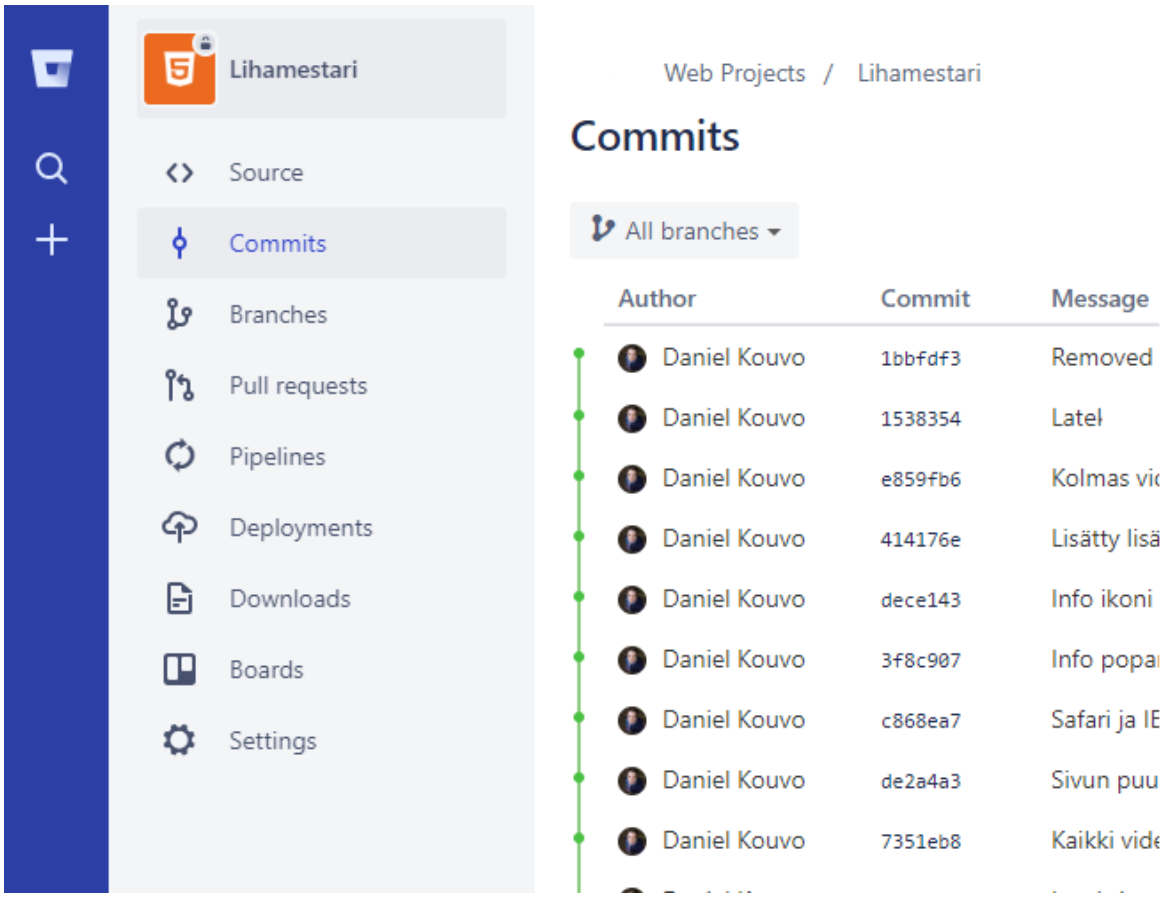
Figur 4. SCSS File Watcher Ruby konfiguration i PhpStorm.

3.4 Bitbucket / Git

Bitbucket är en nätbaserad lagringstjänst för programmeringsprojekt. Bitbucket är speciellt utvecklat för att ett team av programmerare enkelt ska kunna samarbeta i programmerings-projekt. Bitbucket baserar sej på versionshanteringssystemet Git. Elbe (2015) påpekar att en fördel med Git är att man kan gå hur långt som helst tillbaka i historiken.

Hantering av projekt kan skötas enkelt via en webbläsare. Bitbucket innehåller många avancerade och användbara funktioner som t.ex. Webhooks. En Webhook kan t.ex. användas för att ge en s.k. callback till olika tjänster för att uppnå någon slags funktionalitet kopplat med versionshanteringssystemet. I detta projekt har jag bl.a. skapat en Webhook till

automationsservern Jenkins. I Bitbucket får varje projekt en egen Git repository där projektet lagras. Den portabla virtuella utvecklingsmiljön Vagrant och inlärningsspelet finns lagrade som skilda projekt i egna repositories. För inlärningsspelets repository har jag skapat två stycken s.k. branches där den första, Master branch, finns till för en slutlig version av projektet och Staging branch där projektet utvecklas och testas. Jag gör i huvudsak alla mina projekt i Bitbucket (Se figur 5). Därför blev Bitbucket ett naturligt val till det här projektet.

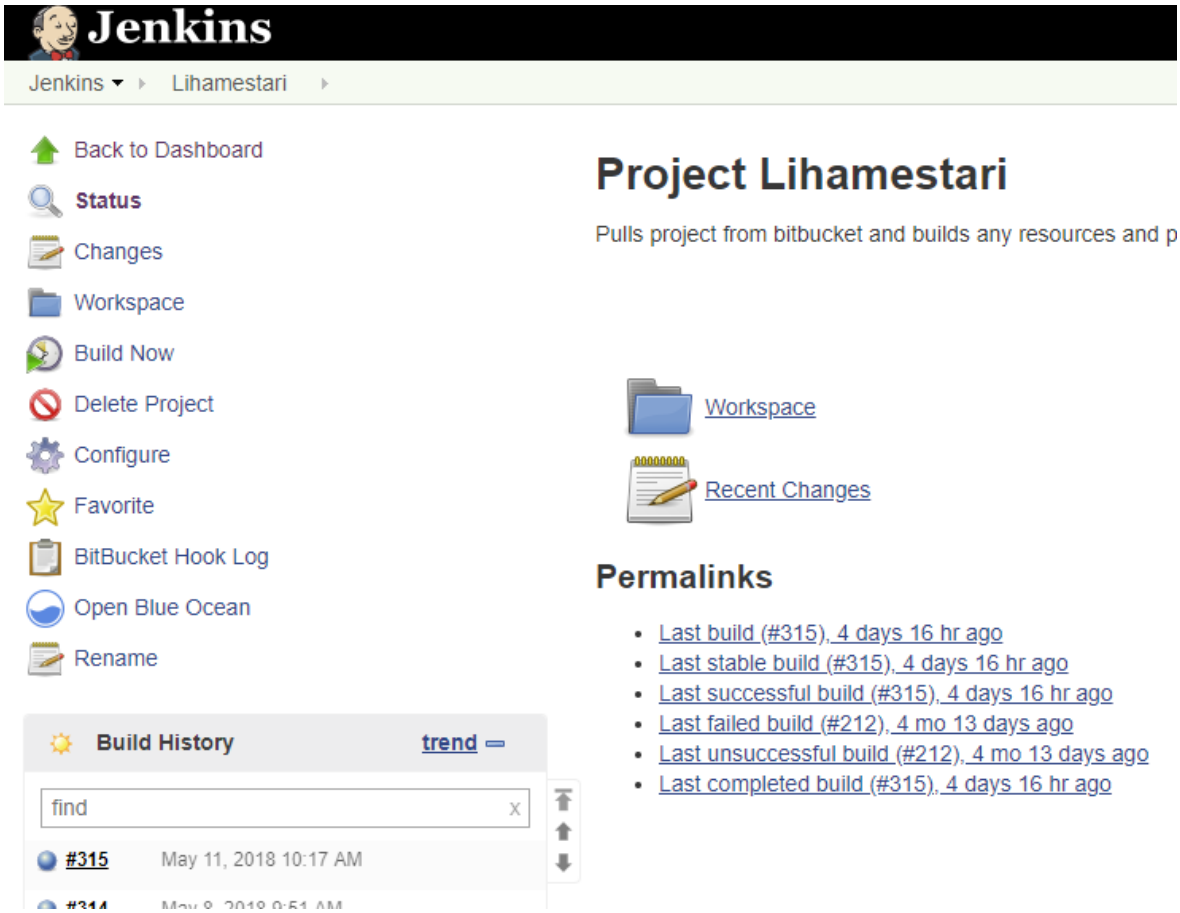


Figur 5. Projektets repository och senaste Commits i Bitbucket.

3.5 Jenkins

Jenkins är en automationsserver baserad på öppen källkod. Jenkins hjälper till med att automatisera skeden som annars skulle göras manuellt, som t.ex. överföring av projektfiler till en webbserver. I detta projekt har jag konfigurerat Jenkins att automatisera förflyttningen av filer från Bitbucket till en webbserver (Se figur 6). Projektets uppdateringar publiceras automatiskt från sin Git repository till en dedikerad Wordpress servermiljö med hjälp av den automatiserade Jenkins servertjänsten. Detta möjliggörs med hjälp av en skräddarsydd Webhook i Bitbucket. Jenkins konfigurationen publicerar automatiskt från rätt Git repository branch till rätt server. T.ex. publiceras allt från Staging branchen till webbservermiljön

avsedd för testning. Tack vare Jenkins automationen har jag kunnat spara in tid i utvecklandet av projektet och samtidigt kunnat eliminera en hel del eventuella mänskliga misstag. Hansson (2015) påpekar att en stor fördel med Jenkins är att det är så enkelt att komma igång med.



Figur 6. Automatiseringsservern Jenkins och projektets senaste build.

3.6 WP Engine

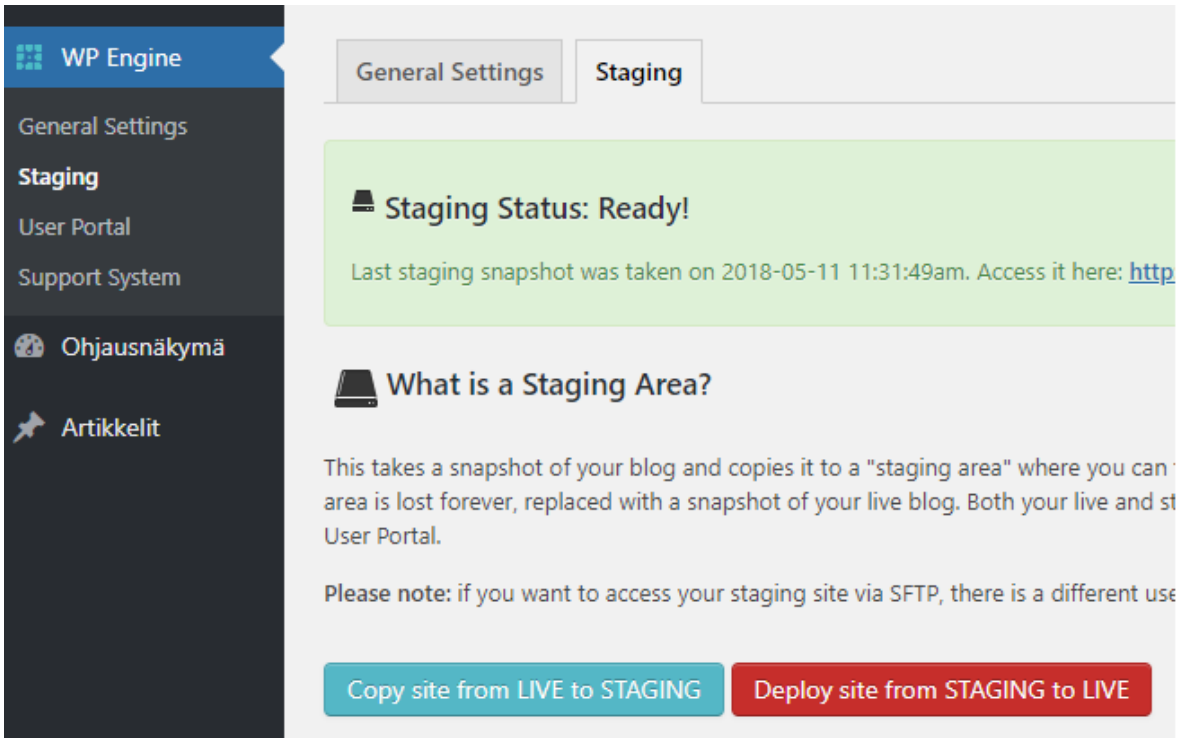
WP Engine är en dedikerad Wordpress servermiljö som har bland marknadens pålitligaste drifttid och utmärkt dygnet runt online kundservice. Det blev klart i början av projektet att inlärningsspelet publiceras i WP Engine servermiljö. Jag har byggt upp många tidigare projekt i WP Engine servermiljö vilket gjorde att detta var en väldigt bekant miljö för mej. BraWebhotell (2014) lyfter följande fördelar med WP Engine. De har finslipat datasäkerheten specifikt för Wordpress. WP Engine erbjuder en avancerad backend för sina kunder där bl.a. en Wordpress installation kan skapas på bara några sekunder. Dagliga säkerhetskopior är standard och även återställning från en backup kan göras bara på några sekunder.

3.6.1 Live

Alla Wordpress installationer och Wordpress baserade webbsidorna körs i en Live miljö i WP Engine. Detta är standardmiljön där en webbsida publiceras för allmänheten, miljön dit en domän konfigureras osv. Vid behov kan även en Staging miljö byggas upp.

3.6.2 Staging

Staging miljön är en miljö som vid behov kan byggas upp. I Live miljön i WordPress egna backend lägger WP Engine till en ”Staging” meny där man via en knapptryckning kan skapa en kopia av Live miljön till en Staging miljö (Se figur 7). Kopian skapas under en minut och är omedelbart i bruk för användning. Jag har valt att alltid skapa en Staging miljö för de projekt jag jobbat med i WP Engine miljö. Det är där jag utvecklar och testar projektet på webben efter den utveckling och testning jag först gjort i lokal miljö. Då ett projekt är klart och testat i Staging miljö är det väldigt enkelt att publicera detta live. Med en enkel knapptryckning i Wordpress backend kan projektet föras från Staging till Live. WP Engine har automatiserat hela processen, även ändring av eventuella sökvägar i databas och kod.



Figur 7. Staging fliken i WordPress i WP Engine miljö.

4 Inlärningsspelet

Detta kapitel redogör arbetsmetoder och olika skeden i uppbyggnaden av inlärningsspelet.

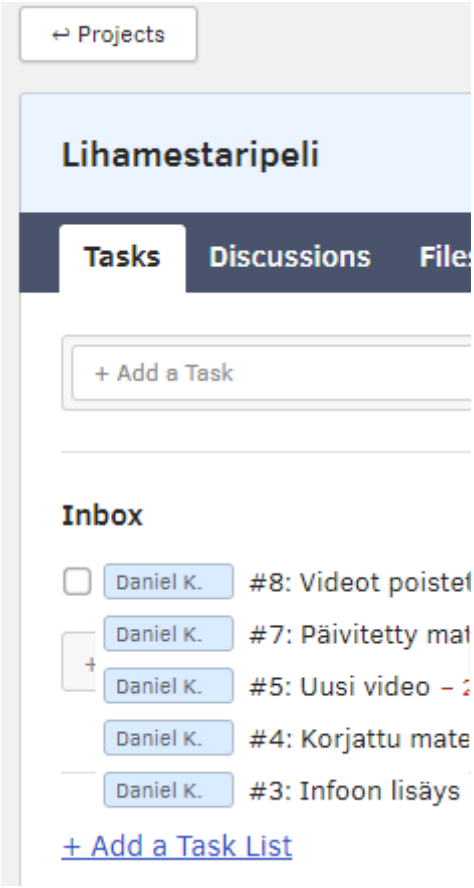
4.1 Arbetsmetoder

Jag har arbetat enligt ett arbetssätt som jag varit van att jobba med från tidigare projekt. I grund och botten handlar det om en agil utvecklingsmetod och en kombination av verktyg och metoder. Projekthanteringsverktyget Active Collab har använts för den interna projekthanteringen där kunden inte direkt har varit engagerat. Trello har använts som ett komplement i projekthanteringen som en plattform där kunden har möjlighet att vara delaktig. Via Trello har kunden kunnat bl.a. ge respons och följa med projektets utveckling. Allt projektmaterial har lagrats strukturerat i Google Drive. En diskussionskanal för projektet skapades inom digibyråns Slack workspace där ärenden relaterade till inlärningsspelet har fritt kunnat diskuteras. Användning av e-post i projekthanteringen och kundkontakten har minimerats så mycket som möjligt. Arbetsmetoderna och verktygen här är även noga utvalda för att möjliggöra att arbetet kan skötas i princip var som helst oberoende dator eller plats.

4.1.1 Active Collab

Active Collab är ett lättanvänt och effektivt projekthanteringsverktyg. Verktyget används via ett webbgränssnitt och stöder även mobil användning. Verktyget är inte låst i någon viss projektmetod men lånar idéer från kända projektmetoder. Varje projekt får en egen avdelning i Active Collab där det är möjligt att skapa s.k. uppgifter (tasks) för projektet. En task kan innehålla beskrivningar, material, en deadline, begränsning på hur lång tid arbetstid det beräknas just för den specifika tasken m.m. En task kan även innehålla s.k. subtasks ifall en task behöver delas upp i mindre delar. Inne i varje task kan arbetstid tillsammans med kommentarer skrivas upp vilket skapar en detaljeras tidsuppföljning för projektet. Detta kan vid behov även kopplas med ett timpris och jämföras mot en projektbudget. Active Collab kan på så sätt hålla koll på ifall ett projekt hålls inom ramarna för en utsatt budget. Rapporter kan skrivas ut från den inskrivna tidsuppföljningen och dessa rapporter kan sedan levereras till kunden i t.ex. Excel format. På motsvarande sätt har även inlärningsspelet skapats som ett projekt i Active Collab (Se figur 8). Projektet har indelats i olika tasks och i dessa har under projektets uppbyggnad en detaljerad tidsuppföljning tillsammans med dokumentation

byggts upp. RedRockHorses (2017) lyfter fram Active Collab som ett av de bästa samarbets- och projekthanteringsverktygen för team.

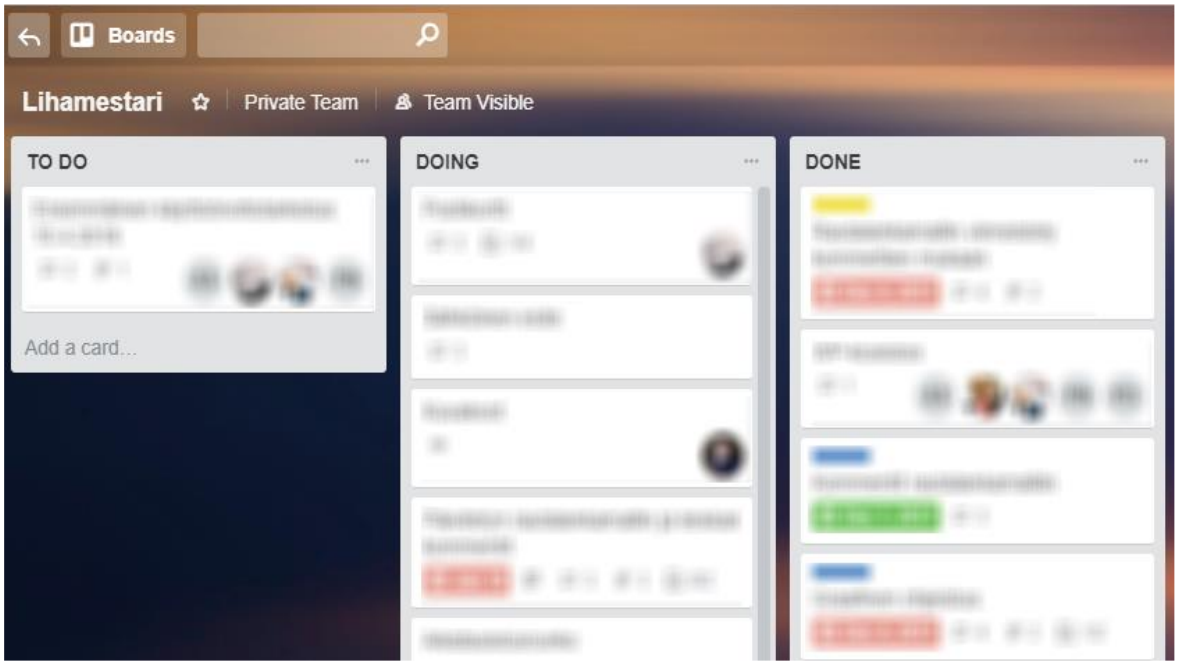


Figur 8. Projektet och tasks i Active Collab.

4.1.2 Trello

Trello är ett populärt, modernt och fortfarande relativt ungt webbaserat projekthanterings-verktyg. Trello har ett lättanvänt och intuitivt gränssnitt där det bl.a. är enkelt att arbeta i team tillsammans med kunder. Trello fungerar som en utmärkt kanal för projektuppföljning där alla parter är delaktiga. Trello gränssnittet påminner till uppbyggnaden om en anslagstavla eller en s.k. Scrum board. I gränssnittet kan ett obegränsat antal olika sektioner byggas upp och namnges valfritt enligt användning, t.ex. ”Att göra”, ”Under arbete”, ”Testning”, ”Gjort”. I sektionerna plockas sedan in s.k. Trello Cards, markeringar som kan vara uppgifter som bör göras eller andra eventuella markeringar. Markeringarna påminner om post-it lappar som limmas på t.ex. en Scrum board. I detta projekt användes Trello inte bara för uppgifter som jag sedan förverkligade utan här fanns även kundens egna markeringar och uppgifter som kunden själv behövde sköta om. Trello blev i detta projekt som ett öppet projektforum med livliga diskussioner och daglig aktivitet (Se figur 9). Tack

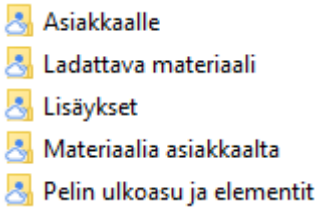
vare Trello kunde eventuell e-post-kommunikation elimineras och kontakten över Trello mellan t.ex. kunden och mej var oftast omedelbar och flöt på smidigt och effektivt. Lagerman (2012) påpekar att allt som görs i Trello händer i realtid och hos alla medlemmar samtidigt.



Figur 9. Projekthanteringen i Trello.

4.1.3 Google Drive

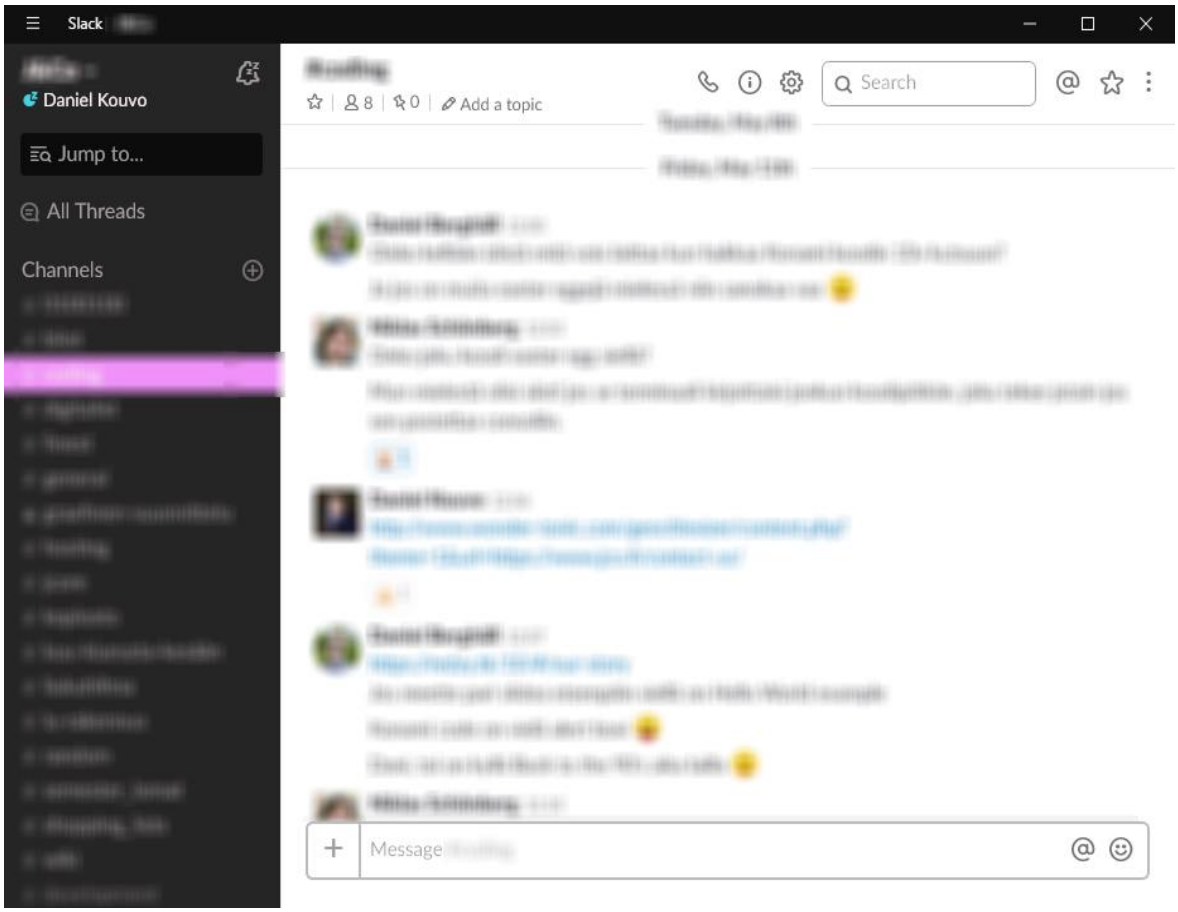
Google Drive är en molntjänst som erbjuder lagringsutrymme av datorfiler över internet och är en slags motsvarighet till Microsoft Office paketet. Allt material till inlärningsspelet som t.ex. grafik och dokument med frågor och svar har lagrats under en specifik projektmapp i Google Drive sorterat i välstrukturerade mappar (Se figur 10).



Figur 10. Projektmaterialet strukturerat i Google Drive, öppnat i Drive File Stream.

4.1.4 Slack

Slack är ett kommunikationsverktyg som i huvudsak är designat som ett samarbetsverktyg för utspridda team. I grund och botten är Slack ett chatprogram som även stöder fildelning, lagring av filer samt ljud- och videokonferens antingen direkt mellan användare i privata diskussioner eller som en grupp i en Slack diskussionskanal. Enligt grundaren av Slack fungerar Slack bra i en miljö som inte är så hierarkisk (Djurberg, Campanello, Nilsson 2017). Slack används som den främsta kommunikationskanalen på den digibyrå där jag arbetat som utvecklare för inlärningsspelet. I Slack skapade jag en egen kanal för inlärningsspelet och bjöd in personer till den kanalen som på ett eller annat sätt hade något med projektet att göra (Se figur 11). Att bjuda in kunden i samma kanal skulle också varit möjligt men eftersom den kommunikationen redan sköttes via Trello såg jag inte behovet att bjuda in kunden till projektkanalen i Slack. Diskussionen på den kanalen var mer för internt bruk. Diskussionen där handlade t.ex. om detaljer för registrering av domännamn för projektet samt annat som berörde projektet.



Figur 11. Diskussion i Slack.

4.2 Planering

Detta kapitel behandlar inlärningsspelets planering, design och uppbyggnad av prototyp. Planeringen startade från en grundläggande idé som kunden levererade elektroniskt. Beställningen gick ut på ett interaktivt webbaserat spel där spelaren svarar på olika frågor och samlar poäng i spelet. Spelet skulle fungera som ett komplement till inlärningsmaterial och handla om köttindustrin i Finland och produktionsprocessen inom den. Frågorna och svaren var informativa och tanken var att genom att spela spelet kan en elev eller studerande lära sej grundläggande information om köttproduktionen i Finland. Jag var ansvarig för att utveckla och bygga upp projektet hela vägen från en prototyp till en slutlig version. Den grundläggande idén från kunden var tydlig men saknade en plan för spelets form och förverkligande. Under ett första planeringsmöte strukturerade vi upp en idé om en bredspelsliknande spelplan där spelaren spelar igenom alla de olika steg som finns inom köttproduktionen genom att svara på frågor och på så vis komma vidare på spelplanen. Idén var att spelaren genom att svara rätt på frågorna samlar på sej biffgafflar som poäng. Baserat på dessa poäng skulle spelaren sedan tilldelas ett diplom i slutet av spelet. Kunden godkände idén och arbetet gick vidare med att jag började förverkliga en prototyp. Grafiken för inlärningsspelet beställdes från en utomstående grafiker.

4.2.1 Prototyp

Jag startade byggandet av prototypen genom att först skapa en Git repository för inlärningsspelet i Bitbucket. Denna Git repository klonade jag till den lokala miljön via PhpStorms versionshanteringsfunktioner. Därefter tog jag ner en Vagrant container som jag även tidigare arbetat med och placerade den i projektet i den lokala miljön. Jag startade upp Vagrant som tack vare sin konfiguration vid uppstart installerar och konfigurerar en virtuell Ubuntu server tillsammans med en WordPress installation. WordPress installationen fanns nu tillgänglig lokalt på adressen vagrant.test där jag via vagrant.test/wp-admin loggade in i WordPress backend och gjorde grundläggande inställningar. Den lokala miljön var nu redo att användas och det egentliga utvecklingsarbetet kunde börja.

Jag inaktiverade det mesta som finns förinstallerat i WordPress, avinstallerade plugins och raderade de förinställda demosidorna, inläggen och kommentarerna. Jag avinstallerade även alla förinställda WordPress teman och startade arbetet med att bygga upp ett skräddarsytt WordPress tema. För att snabbt komma igång med arbetet utgick jag från ett fritt tillgängligt så kallat ”starter theme” baserat på öppen källkod. Detta tema var väldigt avskalat och utvecklat specifikt som en grund för att snabbt komma igång med ett eget skräddarsytt tema.

Jag började med att skala av temat ytterligare genom att t.ex. plocka bort all funktionalitet för WordPress baserade menyer, widgets och annat eftersom de inte skulle ha någon funktion i detta projekt. Därefter startade jag uppbyggnaden av en helt skräddarsydd WordPress sidtemplate som blev grunden för var spelet körs. Tanken med denna sidtemplate var att den genererar en simpel tom vit HTML5 baserad sida med semantisk struktur. All extra funktionalitet som normalt finns med i en WordPress sidtemplate, som t.ex. en PHP loop som hämtar eventuellt innehåll från backend, valde jag helt att skala bort. Idén var att ha ett tomt botten som innehåller en solid HTML5 struktur utan något extra och in i detta botten skulle sedan själva inlärningsspelet renderas.

I det här skedet tog jag itu med själva uppbyggnaden av spelplanen. Grafiken för spelplanen hade levererats till digibyrån från en utomstående grafiker. Spelplanens grafik bestod av en Adobe Illustrator fil, där alla olika grafiska element i spelplanen existerade som skilda element. Från denna design exporterade jag varsamt alla olika element som enskilda PNG filer och förde dem till en specifik bildmapp som jag skapade i inlärningsspelets WordPress tema. Därefter integrerade jag CreateJS i projektet genom att helt enkelt inkludera CreateJS scriptet i WordPress temats head sektion (Se figur 12). CreateJS scriptet aktiverar alla de olika delarna av CreateJS paketet och de finns därmed omedelbart tillgängliga att använda via Javascript.

Figur 12. CreateJS scriptet i WordPress temats head sektion.

Idén var att rendera spelplanen och inlärningsspelet i en HTML5 canvas med hjälp av CreateJS. Jag skapade ett HTML5 canvas element i sidmallen och gav den en unik identitet för att sedan koppla till den från Javascript (Se figur 13).

```
<canvas id="mainCanvas"></canvas>
```

Figur 13. HTML5 canvas elementet skapas.

Canvas elementet visar i sej självt inget annat än en vit ”duk”. För att detta skall vara synligt bör canvas elementet få en bredd och höjd angiven. Jag ville göra canvas elementet så responsivt som möjligt för att stöda olika skärmstorlekar. Utgångspunkten var dock att spelet

alltid används i en horisontell vy. Jag gjorde stilar för canvas elementet som gör att den alltid fyller webbläsarfönstret oberoende skärmstorlek (Se figur 14).

```
#mainCanvas{
    position:absolute;
    width: 100%;
    height: 100%;
}
```

Figur 14. Grundläggande CSS stilarna för canvas elementet.

Med hjälp av Javascript kopplar jag till canvas elementet genom att utnyttja ID:n och lagrar detta i en variabel för att sedan sköta all kommunikation till canvas elementet från Javascript med hjälp av canvas variabeln (Se figur 15).

```
// CANVAS
var canvas = document.getElementById("mainCanvas");
```

Figur 15. Canvas objektet i Javascript.

För de grafiska elementen skapade jag en Javascript array där varje element har en egen id (Se figur 16). Denna array fungerar som ett index med spelets alla grafiska element och skapar ett strukturerat sätt att hämta in och hantera dessa element. Till detta inkluderade jag en variabel som innehåller sökvägen till dessa element.

```
//assets
gameassets = [];
assetPath = "wp-content/themes/peli/assets/";
gameassets.push([
    {src: assetPath + "images/background.png", id: "bg"},
    {src: assetPath + "images/overlay.png", id: "overlay"},
    {src: assetPath + "images/info.png", id: "info"},
    {src: assetPath + "images/help.png", id: "help"},
    {src: assetPath + "images/intro.png", id: "intro"},
    {src: assetPath + "images/start.png", id: "startbutton"}
```

Figur 16. Array för de grafiska elementen som spelet använder.

Jag har utnyttjat CreateJS preloader funktionalitet för att köa upp och ladda in de grafiska elementen. Varje grafiska element blir som ett eget objekt med specifika variabler för varje element. I variablerna lagras även specifika inställningar för objektet som t.ex. värden för x,

y positionering, alpha värde, vilken typ av objekt det är fråga om och t.o.m. eventlisteners kan specificeras i objektet (Se figur 17).

```
//elements
var rect, bg, info, help, intro, text, startbutton, p

function handleFileComplete(event) {
    //background
    if( event.item.id == "bg" ){
        bg = new createjs.Bitmap( event.result );
        bg.alpha = 0;
        lihamestari.addChild( bg );
        fadeIn(bg, 100, 0);
    }
    //info
    if( event.item.id == "info" ){
        info = new createjs.Bitmap( event.result );
        info.x = 360;
        info.y = 100;
        info.alpha = 0;
        info.cursor = "pointer";
        lihamestari.addChild( info );
        info.addEventListener("click", infoPopup);
    }
```

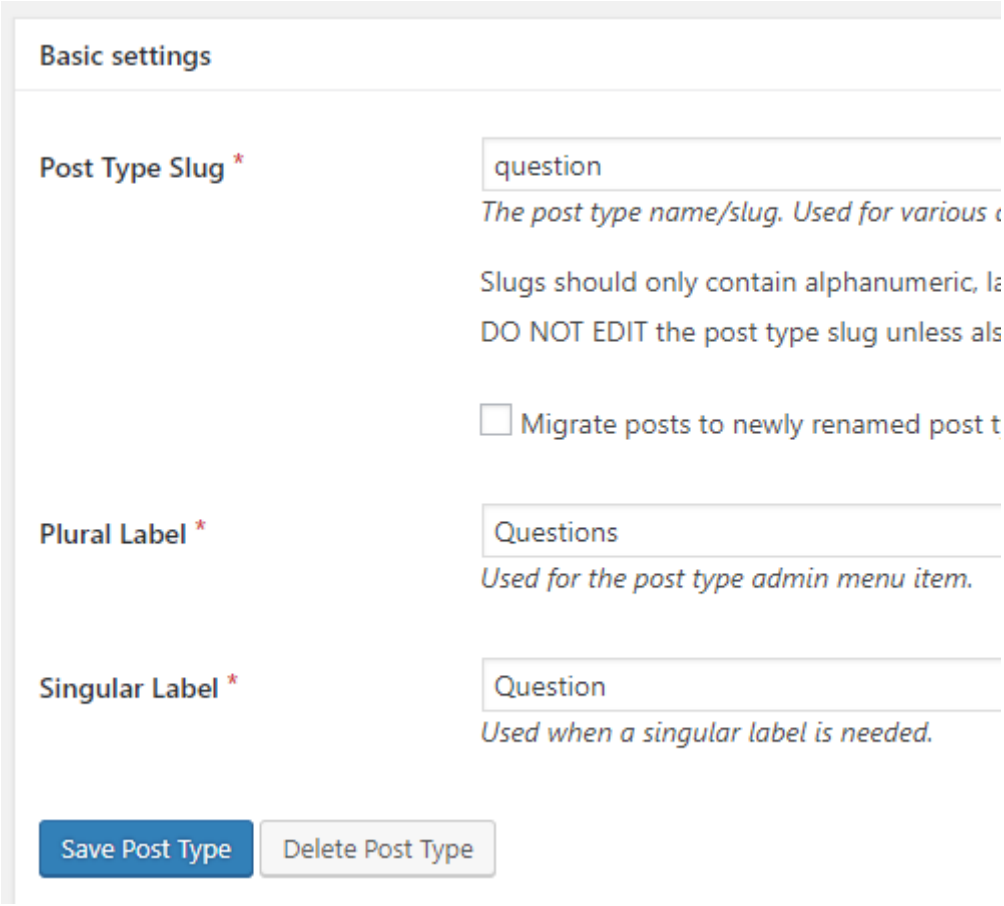
Figur 17. Elementen blir objekt.

Variabeln ”lihamestari” är en CreateJS container som innehåller elementen som kommer att renderas. Slutligen renderas detta i canvas elementet med hjälp av CreateJS.

För prototypen var mitt mål att få spelplanen visuellt uppbyggd i canvas elementet, samt att spelet skulle ha ett fungerande fråge- och svarsystem.

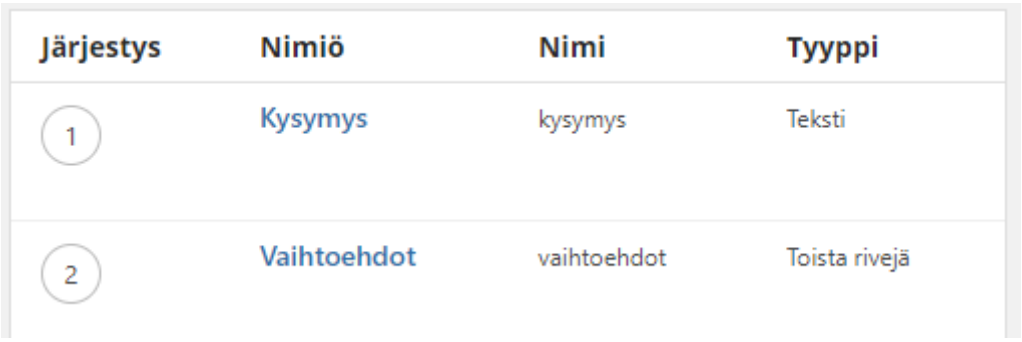
4.2.2 Frågedatabas

Jag skapade en ”Questions” custom post type i WordPress backend som skapade en grund för frågedatabasen (Se figur 18).



Figur 18. En custom post type skapad med WordPress tillägget CPT UI.

Därefter skapade jag skräddarsydda fält för frågedatabasen (Se figur 19). Detta skapade jag med hjälp av Advanced Custom Fields tillägget. Det första fältet består av ett textfält där frågan skrivs in.



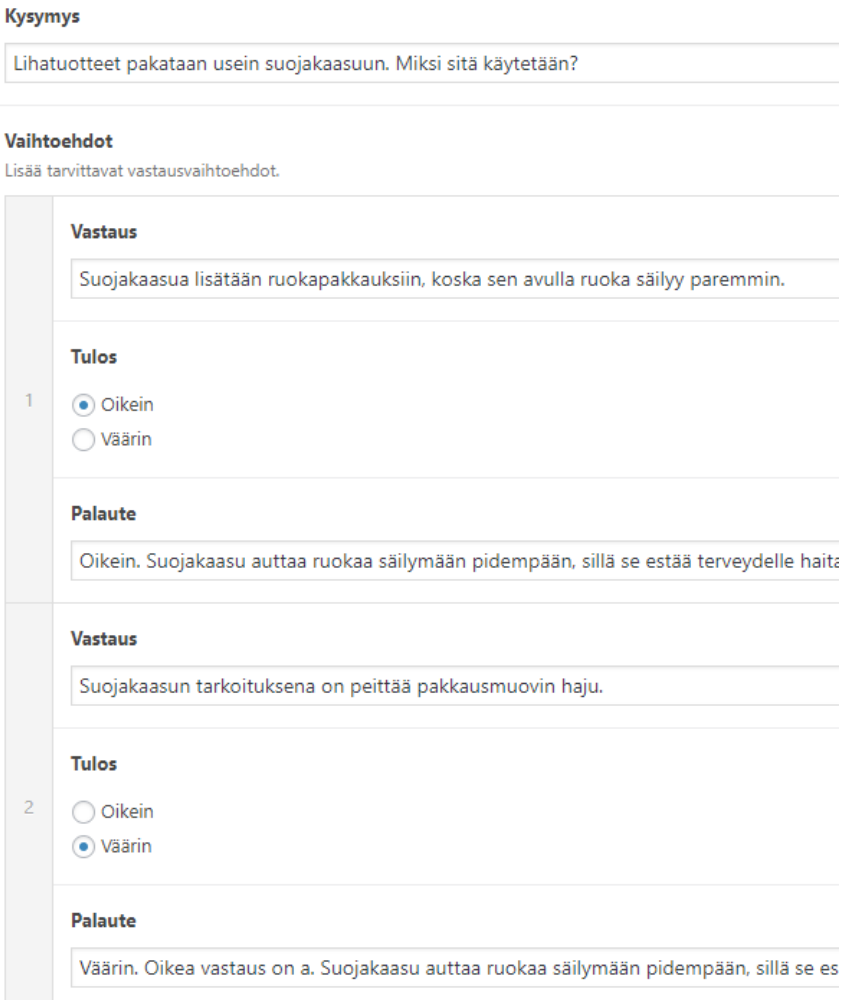
Figur 19. Skräddarsydda fälttyperna för custom post typen ”Questions”.

Det andra består av en s.k. ”repeater field” som kan upprepas och innehåller en uppsättning av specifika fält (Se figur 20). Dessa är ett fält för svarsalternativet, en checkbox för att kryssa ifall svaret är rätt svar och ett textfält för feedback.

Järjestys	Nimiö	Nimi	Tyyppi
1	Vastaus	vastaus	Teksti
2	Tulos	tulos	Valintanappi
3	Palaute	palaute	Teksti

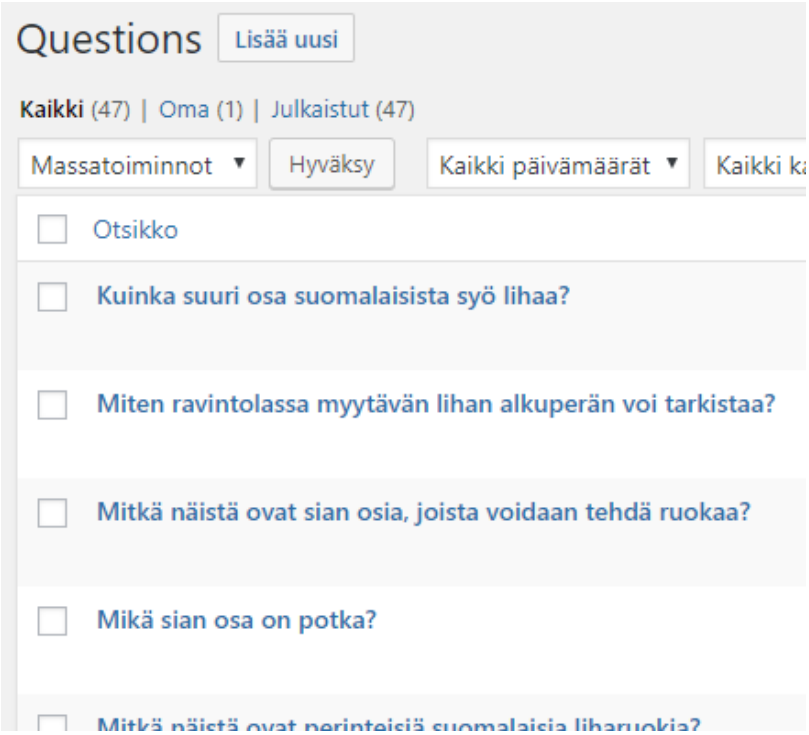
Figur 20. Repeater field fälten som används för svarsalternativen.

Efter detta kunde frågor med svarsalternativ och feedback skapas under ”Questions” custom post typen (Se figur 21).



Figur 21. Ett exempel från frågedatabasen.

Jag matade in alla frågor inklusive svar och resultatet blev en snyggt uppställd frågedatabas där innehåll enkelt kan läggas till, uppdateras och tas bort (Se figur 22).



Figur 22. Frågorna i frågedatabasen.

Jag skapade en REST API baserad hämtning av frågorna och svaren. FSdata (2015) påpekar att REST API möjliggör hantering av data från WordPress via externa tjänster och gränssnitt.

Jag provade även på att skapa en PHP baserad WordPress förfrågan (Se figur 23). Den hämtade alla resultat från frågedatabasen och konverterade dessa till JSON vilket jag sedan kunde utnyttja i Javascript tillsammans med CreateJS. Jag upplevde att den PHP/JSON baserade lösningen fungerade smidigare och gav mej mer möjligheter. Slutligen valde jag den framför REST API hämtningen.

```
if ( $query->have_posts() ) : while ( $query->have_posts() ) : $q
    $answers = array();
    while ( have_rows( selector: 'vaihtoehdot') ){
        the_row();
        $answers[] = array('answer' => get_sub_field( selector: "vast
    }
    array_push( $questions,array('question' => get_field( selector: '
endwhile; endif; wp_reset_query();
?>

<script>
    <?php echo "var questions".$varname ?> = <?php echo json_enco
</script>
```

Figur 23. WordPress förfrågan som hämtar innehållet från frågedatabasen.

Frågehämtningen i spelet hanteras av en funktion som hämtar frågor dynamiskt baserat på parametrar genererat från var i spelet spelaren befinner sej (Se figur 24). Frågetexten infogas i ett CreateJS textobjekt och renderas till canvas elementet ovanpå ett grafiskt element som fungerar som en popup ruta.

```
// Questions
function question(part,nr){
    //change popup text
    popuptxt.text = this["questions"+part][nr].question;

    //show popup
    lihamestari.addChild( popup );
    lihamestari.addChild( popuptxt );
```

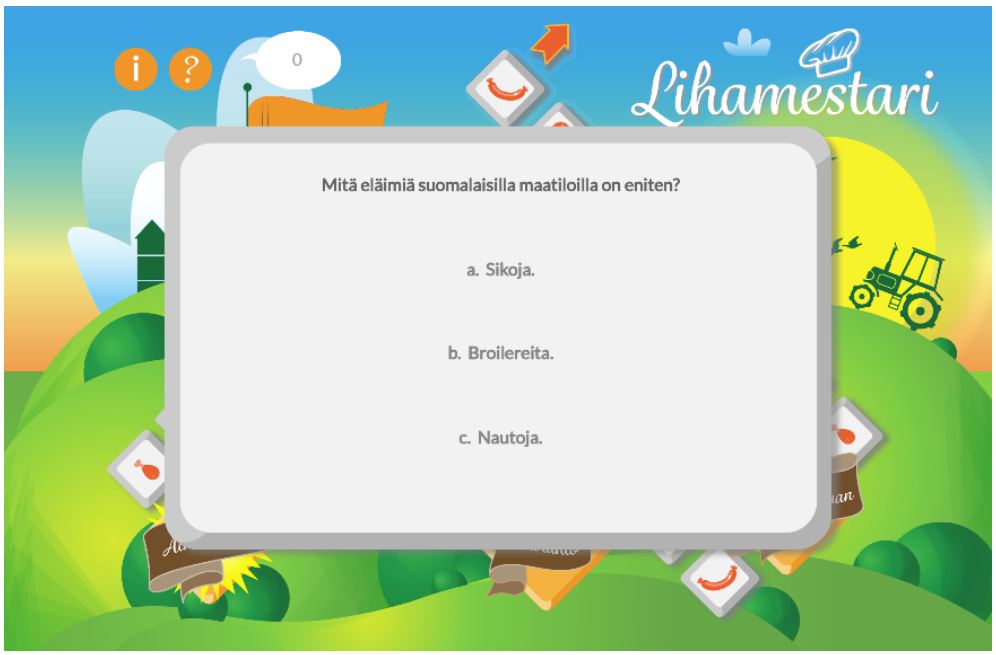
Figur 24. Funktionen som sköter frågorna.

I samma funktion programmerade jag även en loop som bygger upp svarsalternativen samt logiken bakom dem (Se figur 25).

```
// Questions:
for (var i = 0; i < 3; i++) {
    if (this["questions" + part][nr].answers[i]){
        //generate answers a, b and c
        this["answertxt" + i] = new TextLink(String.fromCharCode('a'.charCodeAt() + i)
        this["answertxt" + i].lineWidth = 1480; //line break/word wrap
        this["answertxt" + i].lineHeight = 50;
        this["answertxt" + i].textAlign = "center";
        this["answertxt" + i].x = 1278;
        this["answertxt" + i].y = 600 + i * 200; //increase y position for each answer
        this["answertxt" + i].cursor = "pointer";
        this["answertxt" + i].addEventListener("click", function (event) {
            //generate answer index
            var character = (event.target.text.charAt(0));
```

Figur 25. Loopen som visar svarsalternativen.

Resultatet blev ett fungerande frågesystem som kunden kunde testa i prototypen (Se figur 26).



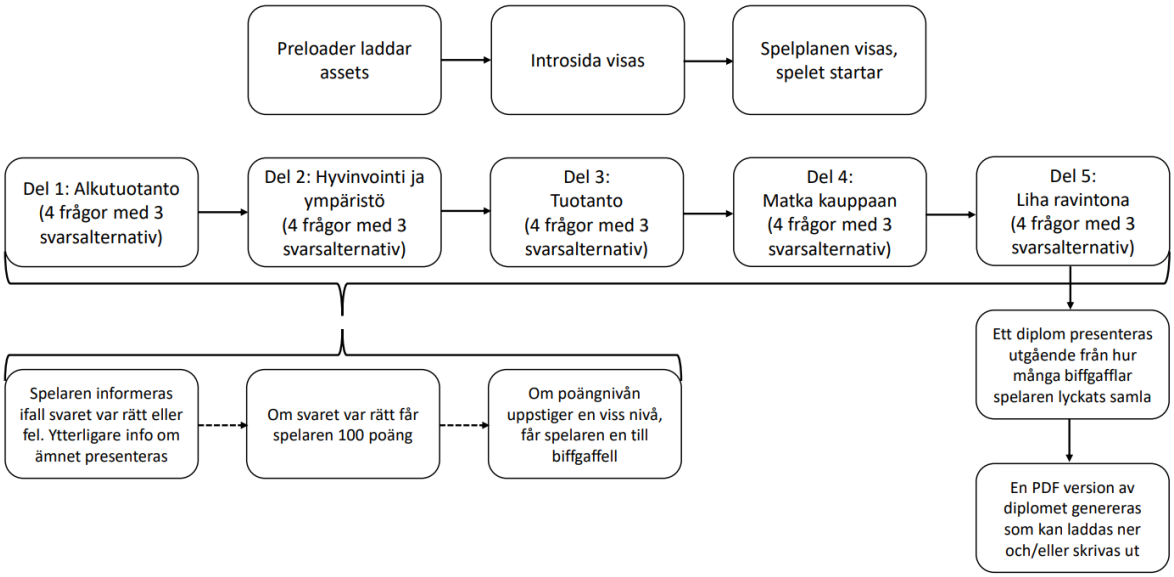
Figur 26. Frågerutan i spelets prototyp.

Under uppbyggnaden gjorde jag regelbundna Git commits tillsammans med kort dokumentation och pushade dessa till projektets repository.

4.2.3 Spelets logik och schema över spelets struktur

Logiken bakom spelet presenteras i nedanstående schema (Se figur 27). En laddningsdialog presenteras som visar att spelet laddar. Därefter visas en introsida med kort introduktion om spelet och en knapp för att starta spelet. Efter att användaren klickat på startknappen, presenteras spelplanen och spelet kan börja. I de fem olika delarna på spelplanen presenteras användaren med fyra frågor per fel och varje fråga har tre svarsalternativ. I de flesta frågor är bara ett svarsalternativ rätt svar medan några av frågorna kan ha flera rätta svarsalternativ. Om användaren svarar rätt berättar spelet med en grön text att svaret var rätt och om svaret är fel berättas detta med röd text. I samband med detta presenteras även mer information om det ämne frågan gällde. För rätt svar får användaren 100 poäng. Poängantalet är synligt på spelplanen. I början av spelet har användaren en biffgaffel. Målet i spelet är att samla så många biffgafflar som möjligt. När poängmängden överstiger 500 får användaren en till biffgaffel. Tre biffgafflar uppnås vid 900 poäng. Fyra biffgafflar uppnås vid 1300 poäng. Fem biffgafflar är maximiantal biffgafflar och uppnås vid 1700 poäng. Högsta möjliga poängantal är 2000 poäng. Efter varje fråga förflyttar sej användaren vidare till nästa fråga och steg på spelplanen tills användaren nått den sista delen av spelplanen. I den sista delen av spelplanen presenteras användaren av ett diplom. Diplomet visar hur många biffgafflar

som uppnåtts samt en text baserad enligt prestationen. Användaren kan fylla i sitt namn i diplomet och generera en PDF version av diplomet som kan laddas ner och/eller skrivas ut.



Figur 27. Schema över spelets struktur.

4.3 Vidareutveckling av inlärningsspelet

Prototypen lade en god grund för spelet och efter ett godkännande och feedback från kunden återstod det slutliga arbetet. Spelet var i det här skedet väldigt statiskt. Alla grafiska element som renderades i canvas elementet öppnade upp utan någon som helst effekt eller animation. Jag aktiverade ticker klassen i CreateJS som fungerar som en slags intervalbaserad rytm vars rytminterval kan utnyttjas till bland annat animering. Ticker klassen har en FPS (Frames Per Second) inställning vars värde anger bl.a. med hur många bilder per sekund något animeras i canvas elementet med hjälp av CreateJS. Därefter byggde jag upp några animerade effekter som kan återanvändas för vilka som helst element. Eftersom allt renderas i en HTML5 canvas kan CSS effekter och animationer inte utnyttjas utan dessa måste i stället programmeras i Javascript med hjälp av CreateJS. Jag skapade en in och ut fade effekt samt en animationseffekt som med slumpmässiga intervaller roterar ett element (Se figur 28). Fade effekterna har jag bl.a. använt för att på ett snyggt sätt bygga upp spelplanen då spelet startar. Vid varje steg på spelplanen finns en bit av kött representerad. Det kan vara en bit korv, en skiva medvurst eller t.ex. en köttklubba. Med hjälp av animationseffekten animeras köttbiten på i det steget som just då är aktivt i spelet. Detta för att bl.a. markera att just det steget är aktivt, samt att användaren ska känna sej intresserad av att klicka på just det steget.

```
//fade out
function fadeOut(canvaselement, time, delay){
    createjs.Tween.get(canvaselement).wait(delay).to({alpha: 0},time);
}

//fade in
function fadeIn(canvaselement, time, delay){
    createjs.Tween.get(canvaselement).wait(delay).to({alpha: 1},time);
}

// lihaanimation
function lihaAnimation(canvaselement){
    var delay = ( Math.random() * 1000 );
    createjs.Tween.get( canvaselement, {loop:true} )
        .wait( delay )
        .to({rotation:-8}, 100, createjs.Ease.quadIn )
        .to({rotation:8}, 300, createjs.Ease.quadInOut )
        .to({rotation:0}, 100, createjs.Ease.quadOut )
}
```

Figur 28. Funktioner för fade effekter och animation.

Biondi (2012) presenterar en del tekniker för att göra CreateJS projekt responsiva. Jag har utnyttjat en del av dessa idéer och utvecklat dem vidare. För att göra spelets canvas mer responsivt och för att få spelet att stöda olika storlekar, skapade jag en eventlistener som känner av ifall webbläsarfönstrets storlek ändrar (Se figur 29). Om den gör det körs en funktion. Denna funktion körs även en gång varje gång då spelet startas.

```
window.addEventListener('resize', setViewSize, false);
setViewSize();
```

Figur 29. Eventlistener som håller koll på om webbläsarfönstrets storlek ändrar.

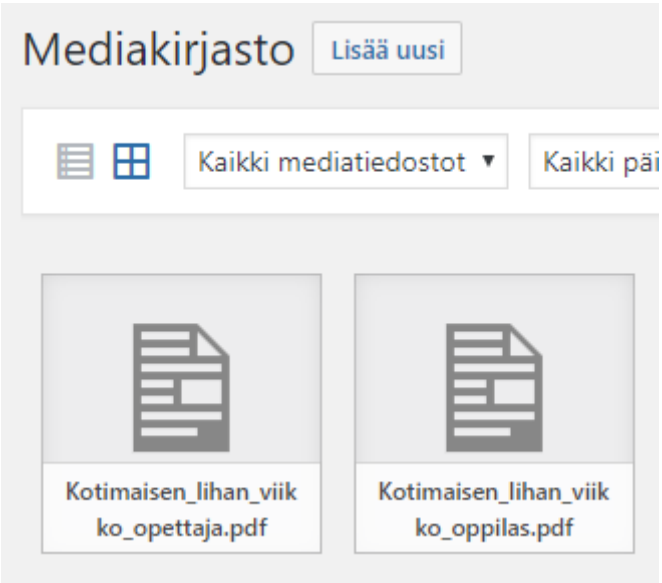
Funktionen ställer in canvas elementets interna bredd och höjd enligt webbläsarfönstrets storlek (Se figur 30). Detta uppdateras varje gång då spelet startas eller om webbläsarfönstrets storlek ändrar. En CreateJS retina funktion används för att bedöma ifall storleken bör skalas upp med det dubbla. En ratio räknas ut för skala grafiken i proportion enligt storlek.

```
//adjust size for display
function setViewSize(){
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;

    if( is_retina_device() ) { // check if retina
        canvas.width = canvas.width * 2;
        canvas.height = canvas.height * 2;
    }

    viewScale = canvas.height / 1536;
    if( viewContainer ) updateView();
}

function updateView(){
    viewContainer.scaleX = viewScale;
    viewContainer.scaleY = viewScale;

    ratio = viewContainer.getBounds().width / viewContainer.getBounds().height;
    var w = ratio * canvas.height;
    viewContainer.x = canvas.width/2 - w/2;
    viewOffset = viewContainer.x;

    stage.update();
}
```

Figur 30. Funktioner för att skala och hålla rätt proportion enligt skärmstorlek.

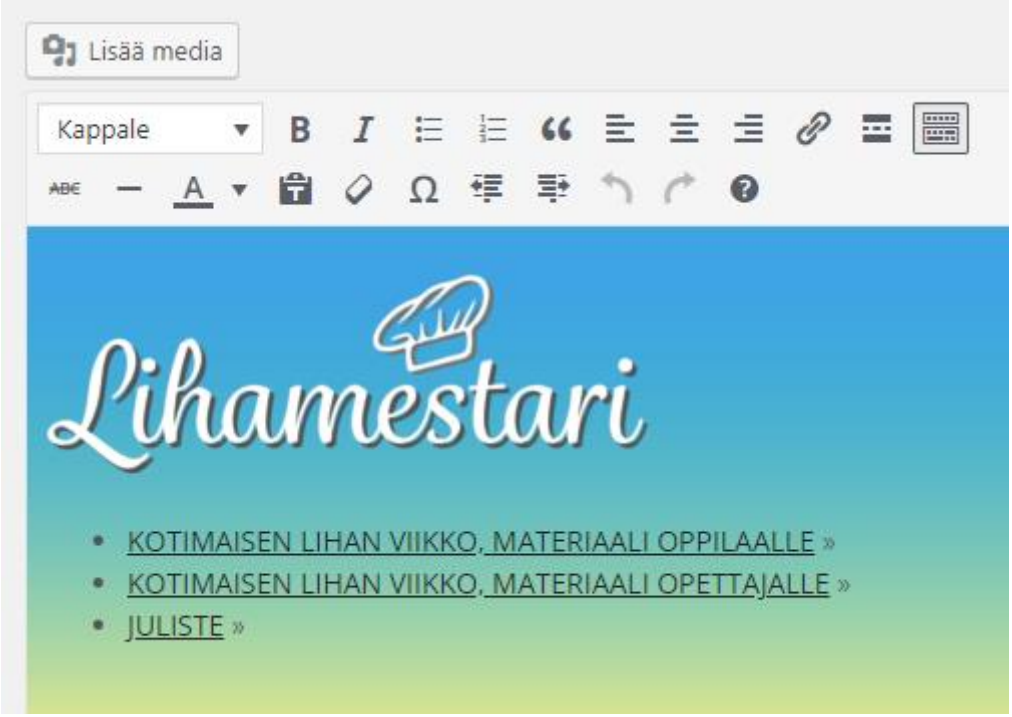
4.3.1 Materialbank

WordPress mediabibliotek utnyttjades som materialbank för att lägga till material ämnade för lärare och studerande (Se figur 31).



Figur 31. Material för nerladdning.

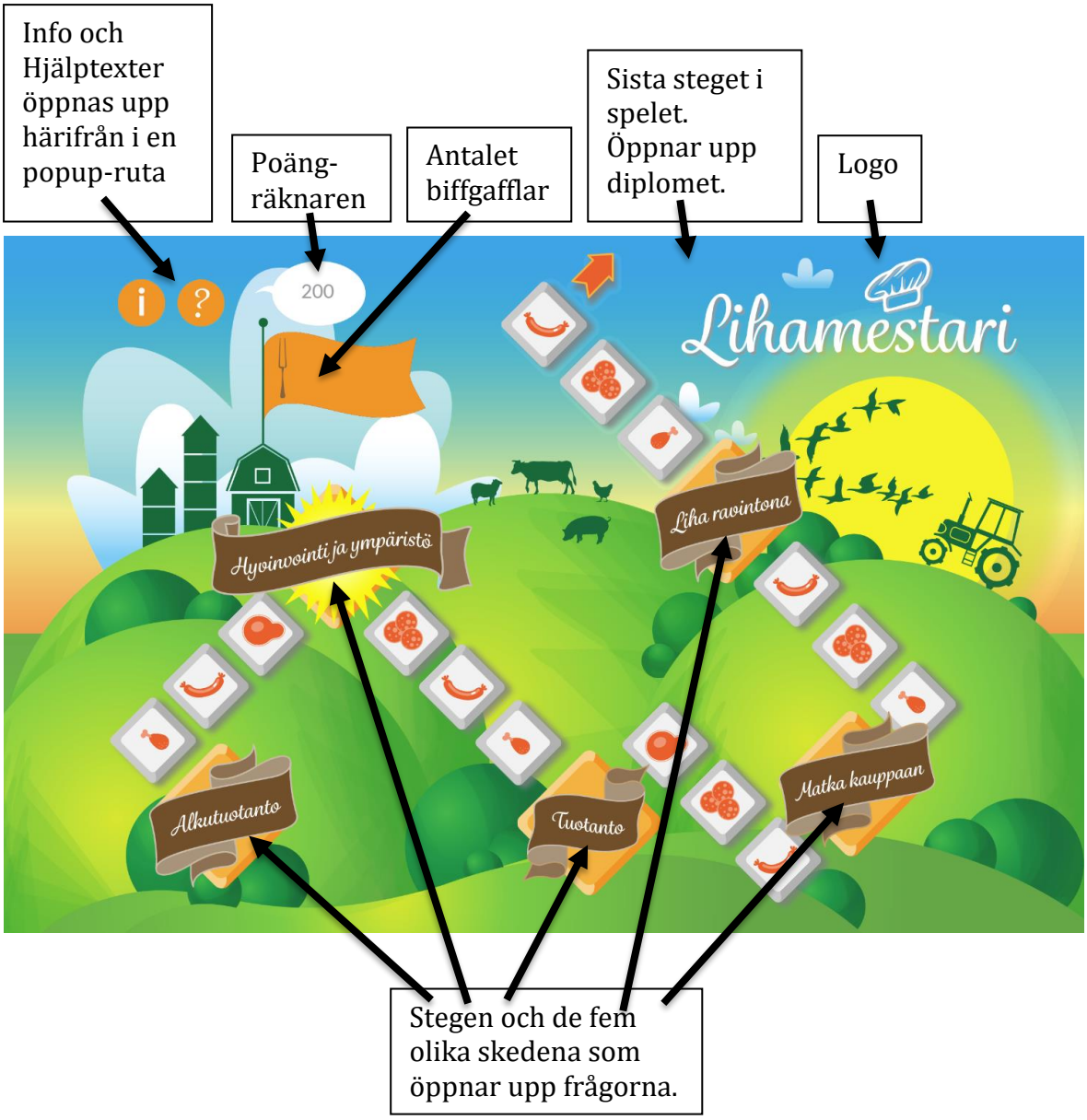
Med hjälp av möjligheten att skapa sidor i WordPress backend, skapades en lösenordskyddad materialsida från vilken lärare kan ladda ner material (Se figur 32). Till denna sida kan material enkelt läggas till och tas bort.



Figur 32. Lösenordsskyddad sida för materialnerladdning.

4.3.2 Spelplan

Nedan presenteras spelplanen och de olika elementen i den (Se figur 33).



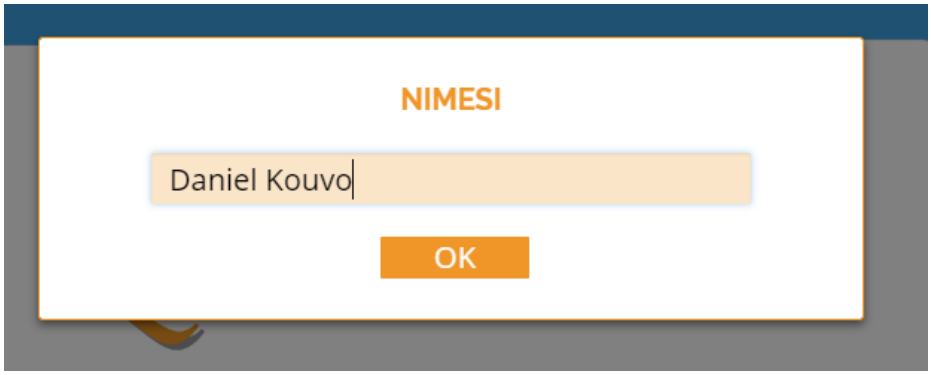
Figur 33. Spelplan.

4.3.3 Ljud

Den slutliga versionen av inlärningsspelet använder inte ljud men stöd för detta byggdes in p.g.a. att det fanns ett önskemål för detta. Detta möjliggör att spelet vid behovs rätt enkelt kan förses med ljud vid ett senare utvecklingsskede. Ljud kan laddas in och läggas till på liknande sätt som de grafiska element som används i spelet.

4.3.4 PDF Diplom

Spelets logik håller koll på hur många biffgafflar användaren samlat och utgående från detta generas ett diplom i canvas elementet. Då detta diplom öppnas, fyller användaren i sitt namn (Se figur 34).

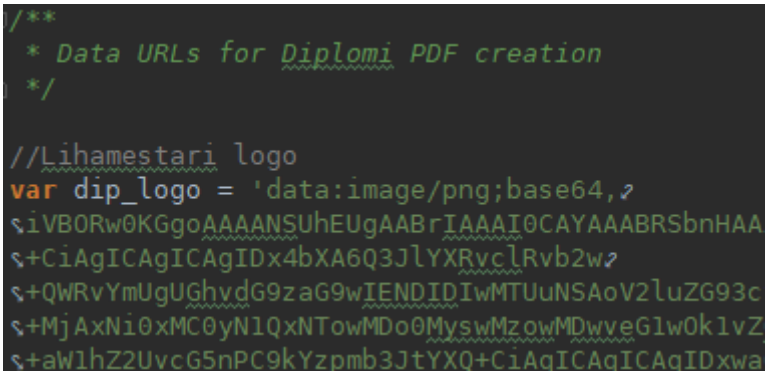


Figur 34. Fält där användaren fyller i sitt namn för diplomet.

Namnet lagras i en variabel och fylls sedan automatiskt in i det genererade diplomet (Se figur 35). Vid behov kan man klicka på namnet och modifiera det ifall man t.ex. gjort ett stavfel.

Figur 35. Diplomet.

För att spara och/eller skriva ut sitt diplom, finns en spara knapp som aktiverar en funktion som automatiskt genererar en PDF av diplomet. Till detta har jag använt Javascript biblioteket jsPDF. Cassandro (2016) påpekar att dokumentationen för jsPDF är rätt bristfällig och presenterar själv olika lösningar för användning av jsPDF. Jag har utnyttjat en del av dessa i min JavaScript funktion. Textinnehållet, namnet och resultatet för diplomet finns lagrat i specifika variabler. Med hjälp av dessa hämtar jag innehållet till den PDF som skapas med hjälp av jsPDF. Eftersom jsPDF inte klarar av att läsa in bildfiler, har jag konverterat all grafik för diplomet till base64 baserad encoding (Se figur 36). Dessa finns lagrade som specifika variabler och kan på så sätt utnyttjas av jsPDF.



Figur 36. Bilder lagrade i variabler som Base64.

Funktionen som skapar PDF diplomet skapar först ett nytt jsPDF dokument (Se figur 37). Till detta dokument byggs innehållet genom att lägga till bildvariablerna, ange fontstorlek, lägga till textinnehållet m.m. Till sist sparas resultatet som en PDF fil.

```
function pdfDiplomi(){
    //Initialize PDF
    var doc = new jsPDF();

    //Set font size
    doc.setFontSize(40);

    //Add Lihamestari logo (imagedataurl, type, x, y, width, height)
    doc.addImage(dip_logo, 'PNG', 29, 20, 150, 49 );

    //Add Name (x, y, text, align)
    doc.text(105, 90, nimi.text, "center");

    //Add Haarukat based on game result
    chooseHaarukka(haarukat); //choose correct png;base64 image
    doc.addImage(dip_haarukat, 'PNG', 75, 110, 60, 36); //add image

    //Change font size
    doc.setFontSize(20);

    //Create multiline text by splitting string into an array
    var splitTitle = doc.splitTextToSize(diplomiteksti[(haarukat-1)], 180);

    //Initial y position of the first line of suoritus text
    var suoritus_line_y = 170;

    //Loop through the array and add multiline Suoritus text
    for(i = 0; i < splitTitle.length; i++) {
        doc.text(105, suoritus_line_y, splitTitle[i], "center"); //add each
        suoritus_line_y+=10; //increase y position for each line
    }

    //Save PDF with custom filename
    doc.save('Lihamestari-Diplomi.pdf')
}
```

Figur 37. Funktionen som skapar PDF diplomet med hjälp av jsPDF.

4.3.5 Loader

Inlärningsspelet består av många olika grafiska element. Att ladda in dessa kan ta en stund och normalt skulle detta innebära att webbläsaren helt enkelt bara laddar på tills allt laddat klart. Användaren kan uppleva det som ett irritationsmoment och det kan kännas som om spelet fastnat. Jag har utnyttjat CreateJS preloader funktionalitet och skapat en dynamisk loader animation som visas innan själva spelet startar. När en användare öppnar upp inlärningsspelet presenteras användaren med en logo för spelet och under den en laddningsindikator (Se figur 38). Det är balk som växer då de olika elementen för sidan laddas. Då den uppnått 100% startar spelet.

Figur 38. Laddningsindikatorn.

5 Sammanfattning

Projektet har gett mej en bra fördjupning i speciellt CreateJS. Jag hade aldrig förut byggt upp ett helt projekt från grunden baserat i stort sätt på CreateJS och nu har jag fått en bra helhetsbild över vad som är möjligt. Att bygga upp ett projekt som helt och hållet renderas i en HTML5 canvas och samtidigt fungerar responsivt var en stor utmaning och nu efteråt ser jag det i stället som en stor möjlighet.

Speciellt intressant har varit att kombinera olika tekniker och system med varandra och att lyckats få dem att kommunicera och fungera som en stabil fungerande helhet. Projektet är väldigt framtidssäkert och möjligheterna för vidare utveckling är enorma.

Att man kan nå en backend där man kan bl.a. hantera innehåll till spelet är en stor fördel för både kunden och en utvecklare.

Att planera och klura ut lösningar för logiken bakom spelet har varit en utmanande och väldigt givande uppgift.

Källförteckning

BraWebhotell, 2016. *WP Engine omdöme* [Online]
https://brawebbhotell.com/sv/wpengine/
[Hämtat 29 April 2018].

Biondi F., 2012. *CreateJS and HTML5 Canvas: resize, fullscreen and liquid layouts* [Online]
http://www.fabiobiondi.com/blog/2012/08/createjs-and-html5-canvas-resize-fullscreen-and-liquid-layouts/
[Hämtat 29 April 2018].

Bång R., 2017. *3 WordPress plugins du inte får missa när du bygger ett tema* [Online]
https://cone.digital/3-wordpress-tillagg-du-inte-far-missa/
[Hämtat 27 April 2018].

Cassandro M., 2016. *Generating PDFs from Web Pages on the Fly with jsPDF* [Online]
https://www.sitepoint.com/generating-pdfs-from-web-pages-on-the-fly-with-jspdf/
[Hämtat 28 April 2018].

Djurberg J., Campanello S., Nilsson S., 2017. *Slack-grundaren i stor intervju: ”Microsoft oroar mig”* [Online]
https://computersweden.idg.se/2.2683/1.691681/slack-tips-samarbetsverktyg
[Hämtat 7 Maj 2018]

Elbe D., 2015. *Vad är Github, git och Bitbucket?* [Online]
http://standout.se/2015/08/12/vad-ar-git-github-bitbucket/
[Hämtat 1 Maj 2018]

FSdata, 2015. *WordPress 4.4 – de stora nyheterna!* [Online]
https://fsdata.se/blogg/wordpress-4-4-de-stora-nyheterna/
[Hämtat 27 April 2018].

Hansson J., 2015. *Kom igång med Jenkins* [Online]
http://www.johnnyhansson.com/Kom-ig%C3%A5ng-med-Jenkins/
[Hämtat 1 Maj 2018]

Lagerman B., 2012. *Vad är Trello?* [Online]
https://trello.com/c/MdeZUUPD/7-vad-%C3%A4r-trello
[Hämtat 7 Maj 2018]

Long J., 2011. *Sass vs. SCSS: which syntax is better?* [Online]
http://www.thesassway.com/editorial/sass-vs-scss-which-syntax-is-better
[Hämtat 17 Maj 2018]

nshalv.com, 2014. *Hur Installera Wordpress lokalt med Vagrant* [Online]
http://www.nshalv.com/hur-till-installera-wordpress-lokalt-med-vagrant/
[Hämtat 29 April 2018].

RedRockHorses, 2017. *Bästa Team Collaboration och Task Management Software för 2017* [Online]
https://sv.redrockhorses.com/app-review/252-best-team-collaboration-and-task-management-software-for-2017.html
[Hämtat 7 maj 2018]

WordPress.org, 2018. *About WordPress*. [Online]
https://wordpress.org/about/
[Hämtat 26 April 2018].