Gruppmedlemmar:

Ansvarsområden:

Henrik: Har ansvarat för dokumentation, hemsidans subsida om funktionell programmering och diverse javakodning med bl.a. livlinor, säkerhetsnivåer och userinputscreen. Skapade frågorna om Java.

Jesse: Har ansvarat för hemsidans indexsida och subsida om reglerna, diverse javakoding som bl.a. livlinorna, endofgamescreen och pristabellen. Skapade frågorna om Javascript.

Joakim: Har ansvarat för att göra grunden till hemsidan, footern på hemsidan, diverse javakodning som bl.a poänguträkningen, återställning av spelet och layout. Skapade frågorna om HTML.

Millad: Har ansvarat för hemsidans aboutsida och diverse javakodning som timerlayout och en del av databaskopplingen.

Mohammad: Har ansvarat för hemsidans subsida highscore, databashanteringen, strukturen i Javadelen och många javadelar som bl.a. frågorna, timern, ljudeffekter, intro och grafik. Har mergat med mastern på GIT. Har skapat UMLdiagram och strukturbilderna i dokumentdelen. Skapade frågorna om CSS.

Spelets idé:

Man ska i Java kunna spela ”Vem vill bli miljonär”. Detta ska ske genom att man i ”frames” via Swing läsa frågor och klicka på rätt svar. Frågorna ska bli svårare och svårare ju längre upp för ”prisstegen” man kommer. Frågorna ska handla om systemutveckling och är uppdelade i fyra delar(CSS, JAVA, HTML och Javascript.

Om man klarar alla 15 frågor har man vunnit.

Spelet har tre stycken ”livlinor” som är ”50/50” där två av de felaktiga svaren plockas bort, ”fråga publiken” och ”ring en vän”.

50/50 kan vi lösa lätt.

Fråga publiken kan antingen lösas genom att med hjälp av random range hjälpa spelaren att gissa på rätt svar där vi gör så att rätt svar kanske visar 65% sannolikhet att just det är rätt svar eller om vi lyckas skapa en funktion via hemsidan där klassen kan ansluta och hjälpa spelaren till rätt alternativ.

Ring en vän är likadant. Antingen ger vi rätt svar med 85% sannolikhet eller så får spelaren kanske fråga Rickard om hjälp på själva presentationen.

Man kan alltid välja att stanna även om man läst frågan. Huruvida vi ska ha säkerhetsnivåer eller inte får vi diskutera.

Svarar man fel förlorar man alla pengar eller åker ner till säkerhetsnivån och får dessa pengar.

Pengarna förs vidare till databasen och ska där sorteras i en ranking som sen ska laddas upp till hemsidan.

Vid lika ska antingen kortast tid ge utslag eller den första som lyckades.

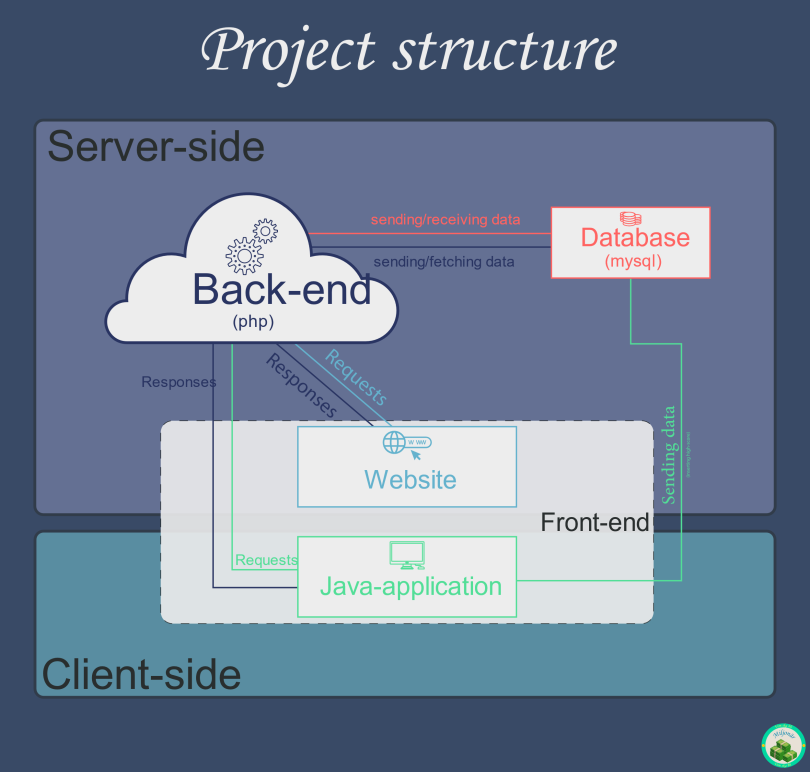
Projektstruktur:  
Projektet består av de följande huvuddelarna:

1. En hemsida som introducerar spelet.
2. En javaapplikation som representerar spelet.

På grund av att hemsidan visar en highscorelista och att spelet behöver många olika frågor skapade vi extra delar som hjälper till att få bra utvecklingsmiljö.  
De två extra delarna är back-end som är skriven med PHP och databasen som lagrar vissa data, exempelvis frågorna som ska användas.

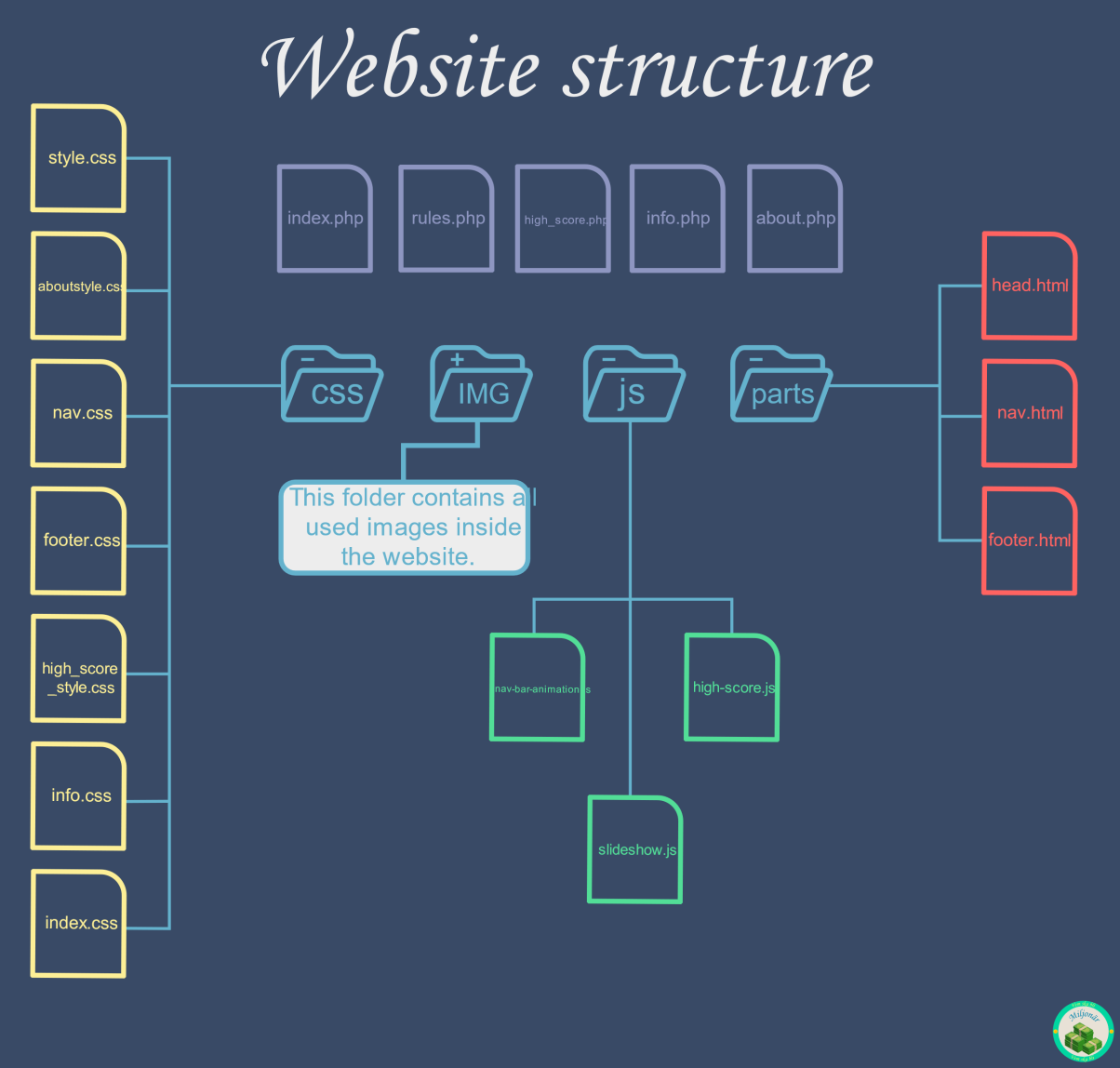
För att hämta in frågarna till spelet skapades en ”restful API” som visar frågorna med sina möjliga alternativ i JSON format istället för att ha direktkontakt mellan javaappen och databasen.  
Syftet med det är att man inte behöver utveckla en integrerad back-end i java som hämtar frågorna och sen en annan om vill ha webbaserad version av spelet.

Databasen är en mysql databas och innehåller de två följande tabellerna:  
- questions: lagrar frågorna och sina alternativ som visas till spelare.  
- highscore: lagrar spelares spelomgång.

Följande diagram beskriver projektets struktur. ”Vissa texter är inte på plats på följande diagram. öppna de själv stående diagram ” SVG filer” för att ha bättre resultat.”

Hemsidans struktur:

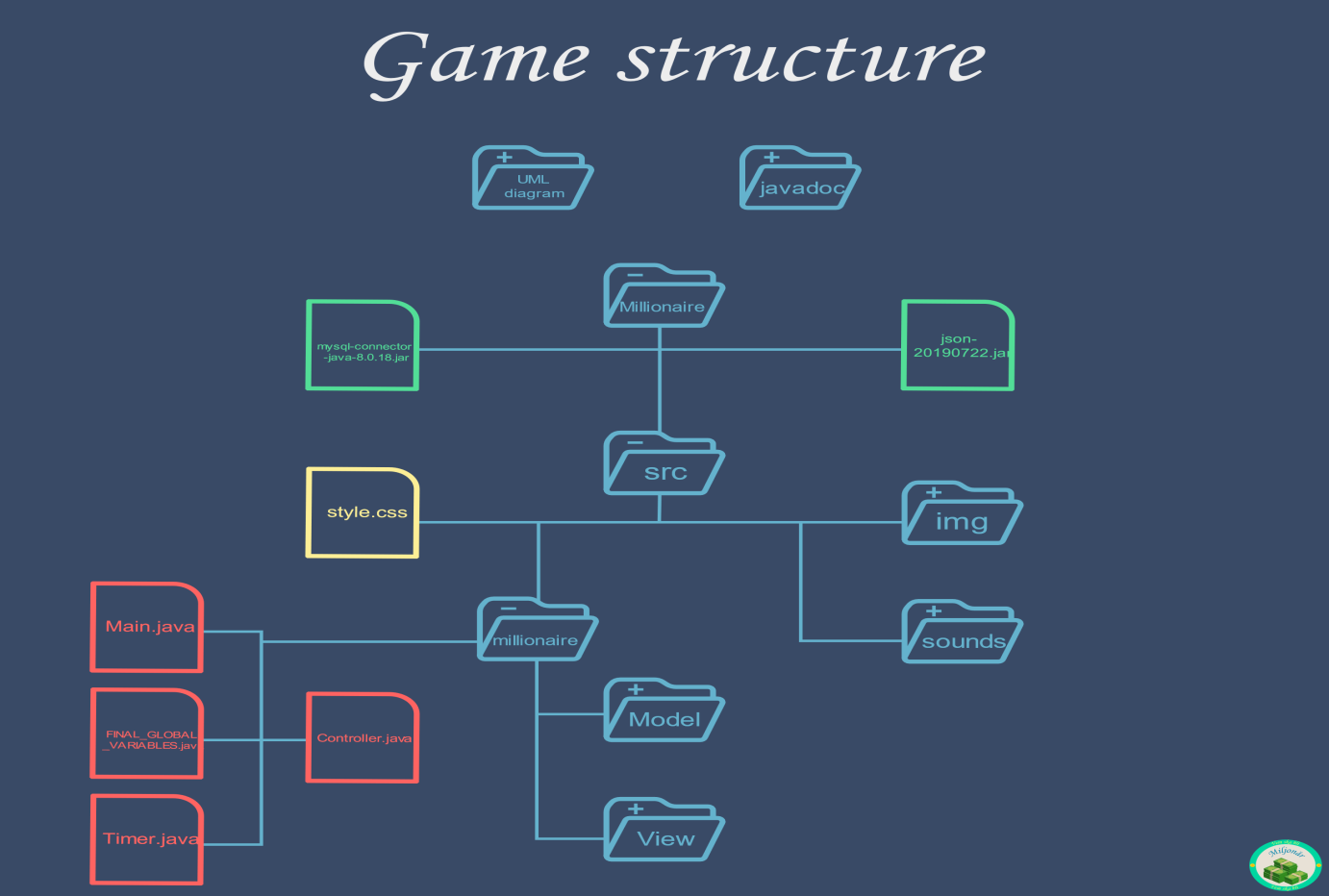
Följande diagram beskriver hemsidans struktur:



* index.php: visar hemsidans huvudsida.
* rules.php: visar Regler sidan.
* high\_score.php: visar High-score lista sida.
* info.php: visar Funktionell programmering sida.
* about.info: visar Om Oss sida.
* CSS : mappen innehåller alla CSS-filer som används för att styla hemsidan.
* IMG : mappen innehåller alla bilder som används på hemsidan.
* js : mappen innehåller alla javascriptfiler som används på hemsidan.
* Parts : mappen innehåller alla hemsidans små komponenter, exempelvis navigation bar och footern som visas i alla subsidorna.

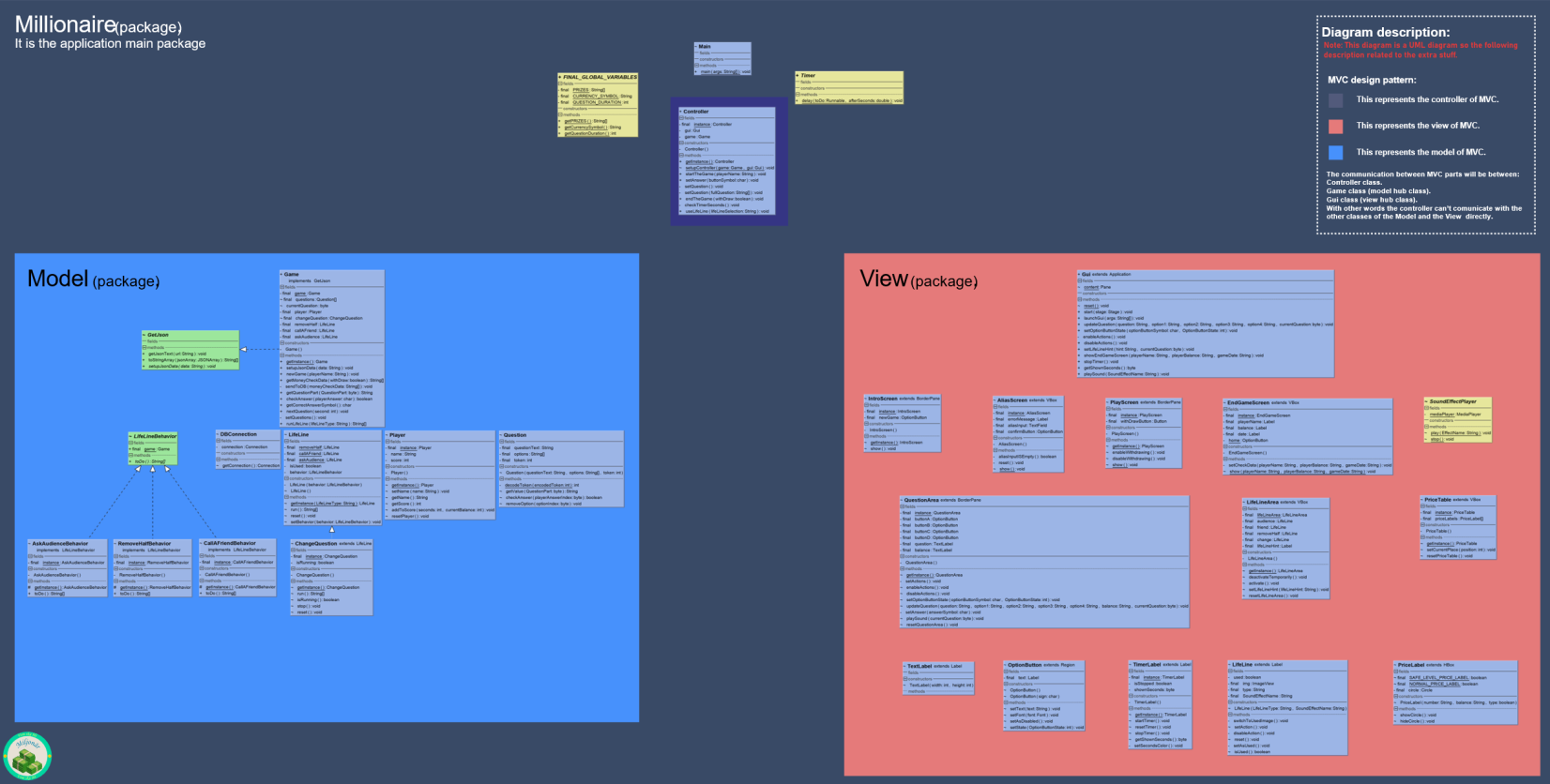
Spelets struktur: (javaappen)

Följande diagram beskriver spelets struktur:



* UML diagram : mappen innehåller appens UML diagram.
* Javadoc : mappen innehåller appens javadoc.
* Millionaire: mappen innehåller externa bibliotek som användas för att behandla JSON och skaffa mysql-anslutning samt srcmappen.
* src : mappen innehåller mapparna img, sounds, millionaire samt style.css filen.
* img : mappen innehåller alla bilder som används i spelet.
* sounds : mappen innehåller alla audiofiler som används i spelet.
* style.css : har CSS koden som stylar vissa komponenter.
* millionaire : mappen innehåller alla javaklasser.
* Model: mappen innehåller alla klasser som tillhör delen ”Model” av MVC.
* View: mappen innehåller alla klasser som tillhör delen ”View” av MVC.
* Controller.java : klassen som representerar delen ”controller” av MVC.
* Main.java : är appens huvudklass.
* Alla andra klasser är beskrivna i javadoc.

Följande UML diagram visar appens klasser och sina innehåll:

Appens designmönster:  
Följande designmönster används i appen:

* Singleton.
* Strategy.
* Model View Controller (MVC).

Singleton:  
Används av vissa klasser som inte ska skapa fler än en instans.  
Exempelvis:  
Game klassen i Model eller PlayScreen klassen i Viewen.

Strategy:  
Används av vissa klasser som är lika men har olika beteende ”Strategy”.  
Exemplevis:  
De tre Klasser som implementerar LifeLineBehavior interfacet.

MVC:  
Används som ett huvuddesignmönster som separerar appens delar så det blir lättare och bättre att underhålla och utveckla appen vidare om man vill.

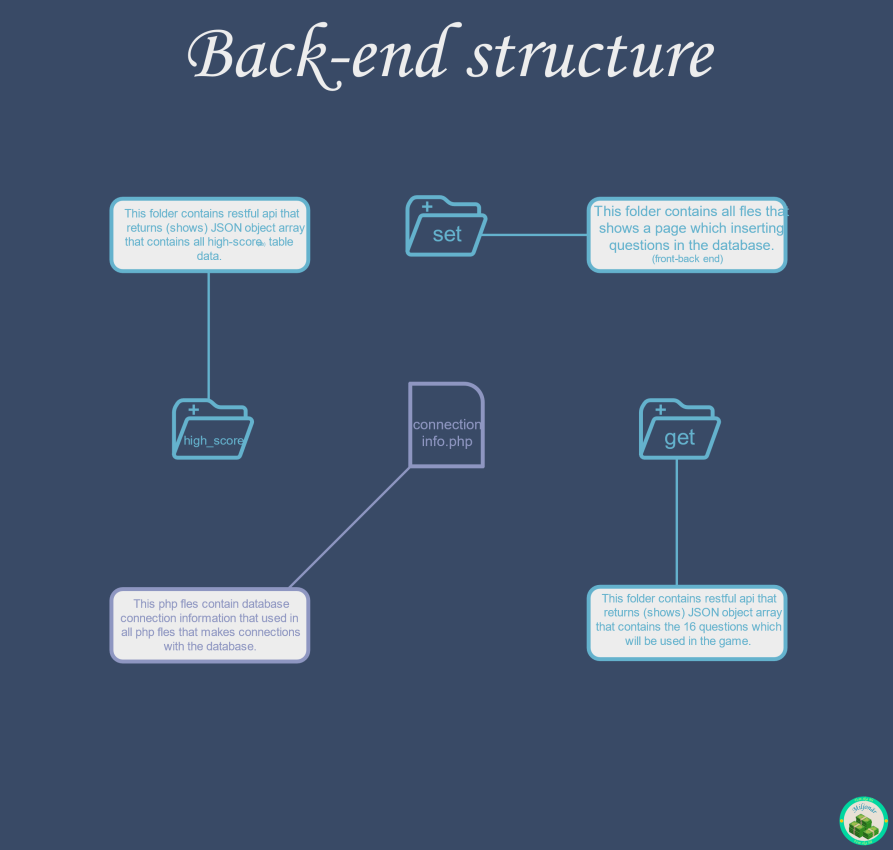
Vi utvecklade MVC så att man inte behöver många variabler i Kontrollern för att skilja ankommande event. Alltså använde vi inte ”event handler” utan använde lambdaexpression för att anropa vissa metoder direkt istället.

Om vissa saker bara ändras i view när man tex trycker på en knapp ska kontrollern inte blandas in i utan det ska bara tas hand om i view. Men om några ändringar behöver data av modellen ska kontrollern ta hand om det. Dvs skicka signaler till view med datan som behövdes efter att ha hämtat datan från modellen.

Viewen har ingen aning vad modellen har för klasser och vice versa.  
Kontrollern kommunicerar bara med modellens huvudklass (Game) av klasserna i modellpaketet.  
Kontrollen kommunicerar bara med views huvudklass(Gui) av klasserna i viewpaketet.

Kontrollerns största roll är att vara en koppling mellan viewen och modellen.

Back-end struktur:  
Följande diagram beskriver back-end struktur:



Detta diagrammet beskriver back-enden tillräckligt.  
OBS! Alla ovanför filar och mappar måste stå tillsamman med hemsidans filar alltså under samma mappen.

Backlogen:

Javadelen:

* Leta upp ljudfiler.
* Börja bygga grunden.
* Spara highscore för att sedan kunna lägga till i databas.
* Kommentera för JavaDoc.
* Arv polymorfism.
* GUI, Swing.
* Versionshantering GIT.

HTMLdelen:

* Funktion för att läsa från databasen och presentera datan. MÅSTE använda AJAX.
* Funktion för att läsa reglerna och förstå hur highscorelistan konstrueras.
* Del som beskriver vad funktionell programmering är.
* Dynamsika webbapplikationer med serversidespråker PHP.
* Versionshantering GIT.

UX:

* Göra hemsidan responsiv.
* Ha bra design på spelet och hemsidan.

Databasen:

* Skaffa db grunden.
* Skaffa addera ny fråga delen (front+back)
* Utveckla addera ny fråga delen
* Skaffa php sidan för att hämta spel frågor.
* Skaffa highscore delen (back).

Spelplanering:

* Kortfattat vad spelet ska gå ut på
* Komma på 15 frågor per person om varsitt ämne
* Inlämning/Github:
* Projektets gång
* Inlämning backend
* Inlämning frontend
* Inlämning Worddocument
* Inlämning powerpoint
* Dubbelkolla kraven som Rickard gett oss

Documentdelen:

* Organisera Trello
* Skapa Messengergrupp
* Veckovis log: Vem , vad?
* Backlog(Initiala versionen)
* Förklaring av spelet
* Gruppmedlemmar
* Ansvarsområde för varje medlem
* Struktur av spel: klasser, metoder, variabler. Ska visas med Bilder och text.
* Varför valde vi våra datastruktur. Arrays, list?
* Varför valde vi och hur ser vårat designmönster ut? MCV?
* Struktur av hemsida.
* Vilken säkerhet har vi och vilken säkerhet skulle vi vilja ha?
* Vilka funktioner skulle vi vilja ha om det fanns mer tid?
* Utvärdering. Bra, dåligt. +-!?. GIT?
* Vad är algoritmer?
* Redogöra för versionshantering, GIT
* Vad är CMS?
* Vad är funktionell programmering?
* Vad är tex XSS(Cross-site Scripting). Säkerhet i script.

Presentation:

* Gå igenom dokumentets punkter.
* Demonstrera projektet.
* Fördelar/nackdelar med CMS.

Veckovis utveckling:

Vecka 1:

* Alla skapade sina frågor och laddade upp dessa på databasen som Mohammad skapade.
* Joakim började skissa på hemsidan och Jesse på javadelen.
* Henrik tog tag i organiseringen.

Vecka 2:

* Millad anslöt sent till gruppen och denna vecka bestämde vi att vi skulle använda JavaFX till projektet så alla läste på om detta.
* Mohammad lade upp projektet på Github och alla anslöt sig till detta.
* Mohammad utvecklade Jesses javagrund och kopplade denna till databasen.

Vecka 3:

* Denna vecka var ren javakodning där alla gjorde metoder till spelet och spelet tog form ordentligt.

Vecka 4:

* Mohammad skapade alla bilder som Henrik satte på plats.
* Jesse fick ordning på pristabellen.
* Mohammad slutförde så att frågorna och svarsalternativen kom på plats.
* Samtidigt arbetade allihopa med metoder i Java.

Vecka 5:

* Javakodningen fortlöpte när alla utom Mohammad jobbade med varsin ledtråd.
* Mohammad väntade in alla andra och arbetade då parallellt med en potentiell webbversion av spelet.

Vecka 6:

* Mohammad hjälpte de som kört fast för att gruppen kände att det börjar bli dags att sätta spelet och att vi vill ha klart det nästa eller nästnästa vecka.
* Mohammad färdigställde även timern och ljudeffekterna.
* Henrik gjorde klart userinputscreen och Jesse endgamescreen.
* Millad tog tag i hur spelet skicka spelomgången till databasen.
* Vi gjorde även regler för vad som ska gälla för gruppen. Inte för att det hade hänt något utan för att vi glömt det till en börja.
* Vi planerade även hur hemsidan ska se ut. Denna tog återigen Joakim tag i och började designa denna.

Vecka 7:

* Spelet var nu spelbart men det återstod många buggfix och metoder för att få det som vi ville.
* Millad gjorde en grafisk detalj med timern och arbetade sen med hemsidans aboutsida.
* Joakim löste metoden som räknar ut playerscore och grafiska detaljer med texten.
* Jesse började med subsidan regler.
* Henrik tog tag i dokumentdelen och subsidan funktionell programmering.
* Mohammad fick designa om strukturen då vi upptäckt att denna behöver justeras för att få ett så bra designpattern som möjligt. Han började även på subsidan highscore och hur denna ska läsa in datan från databasen.

Vecka 8:

* Alla hade arbetat med sin HTMLdel.
* Millad hade arbetat med aboutsidan.
* Henrik med funktionell programmering
* Jesse med regelsidan, Joakim med footern och Mohammad med highscoresidan.
* Mohammad hade även strukturerat om klasserna.

Vecka 9:

* Alla hade under jullovet arbetat med sin del i javadoc och sin del av hemsidan.
* Mohammad hade gjort diverse diagram och skrivit texten som tillhör dessa(projekt struktur).
* En del buggar har även fixats i spelet.

Vecka 10:

* Presentation!

Säkerhet:

Planerade utvecklingar:

Utvärdering av projektet:

Det vi gjorde bra var att vi klarade av det vi verkligen ville göra. Vi samarbetade till slut bra och om någon körde fast hjälpte vi varandra istället för att projektet stod still. Vi kommunicerade bra på messenger och använde Trello och GIT.

Det vi kunde gjort bättre var att redan från början tänka ut designpatterns och hur strukturen skulle vara. Detta saknade vi dock kunskap om då. En annan sak vi kunde ha gjort är att från början skriva ner alla klasser och metoder som skulle behövas men det hade varit svårt och något sätt var det denna gång bra att addera klasser och metoder allt eftersom.

GIT fungerade till slut ganska bra. Det var svårt i början men ju mer träning man fick desto bättre gick det. Man lärde sig även med tiden att skriva bra commitkommentarer och att inte committa för ofta. Då detta var första gången vi använde GIT och alla var osäkra så arbetade vi på det sättet att alla pushade upp sin egen branch och Mohammad mergade alltid dessa med Master branch. På så sätt minimerade vi konflikter i koden.