Rapport:

Inlämningsuppgift u05-car-rental

## Syfte

Uppgiften är att utveckla en applikation med hjälp av PHP, JavaScript och Twig som beskriver ett system för uthyrning av bilar. Applikation skall baseras på MVC-modellen och kommunicera via en router. Applikationens data skall lagras i en databas med hjälp av MySQL.

Jag har valt att avvika något från den struktur på uppgiften som framgått av den delade specifikationen. I stället för att ha en speciell sida för huvudmeny har jag valt att integrera denna på samtliga sidor. För mig framstår detta som betydligt mer användarvänligt då det undviker onödig navigering. Jag har också gett användaren möjlighet att ångra gjorda raderingar från kund- och biltabellerna. Tyvärr har jag inte haft tid att göra så att en sådan ändring även backar gjorda rensningar i uthyrningstabellen. För detta skulle samtliga ändrade rader behöva lagras tillfälligt för att kunna återställas. Det är inget stort problem men jag har inte gjort det.

## Arkitektur

Applikationen baseras på en .htaccess fil som styr om all trafik till en och samma index.php fil. I denna index-fil skapas en instans av klassen Router som med hjälp av ett objekt av klassen Request analyserar den sökväg som angetts i URI:n. Routern bryter ner sökvägen i sina beståndsdelar vilka genom regex-mönstermatchning paras ihop med möjliga sökvägar lagrade i en separat fil. Genom att matcha sökvägen mot de fördefinierade vägarna styrs anropet till det Controller-objekt (kontrollanten) som ansvarar för denna typ av anrop och den aktuella metoden på kontrollanten anropas.

Den åberopade metoden på kontrollant-objektet ansvarar för eventuell applikationslogik och att, vid behov, kommunicera med datalagret för att därefter vidarebefordra data att presentera skärmen. Själva presentationen sker genom Twig. Kommunikationen med datalagret sker genom separata modeller som åberopas från kontrollanten.

Kontrollanterna kommunicerar, som sagt, inte direkt med databasen utan med modeller, klasser, som ansvarar för att i sin tur skriva, modifiera, ta bort och hämta data från databasen. Då de olika kontrollerna gör liknande anrop har applikationen en super-modell-klass som de andra modellerna ärver från. Det är denna superklass som ansvarar för kommunikationen med databasen medan underklasserna ansvarar för att fungera som ett interface gentemot kontrollerna. På så vis undviks repetitiv kod och genom underklasserna kan relativt komplicerade metodanrop döljas från kontrollerna.

Datalagrets modeller, vilka är egna klasser, motsvarar M (Model) i MVC-modellen. De presentationer som görs av Twig gentemot användaren motsvarar V (View) i MVC-modellen. Slutligen motsvarar kontrollanterna C (Controller).

## Dokumentation

Applikationen är utförligt kommenterad vilket får tjäna som den primära dokumentationskällan. Här kommer en översiktlig genomgång av beståndsdelarna. Denna genomgång kommer att följa den befintliga mappstrukturen.

En bild som visar skärmbild

Automatiskt genererad beskrivning

Config-mappen innehåller en config-fil med inloggningsuppgifter för databasen samt applikationens ”bas-url”. Bas-url används för att applikationen på min server inte ligger i rot-foldern utan i en under-katalog. I config-mappen finns även filen, routes.json, som innehåller applikationens sökvägar och som används av routern.

Deploy-mappen innehåller modifierade versioner av config-filen och routes.json-filen för den version av applikationen som ligger uppe på Internet. Dessa filer versions hanteras inte av säkerhetsskäl.

I img-mappen finns bara en bakgrundsbild som visas på startsidan.

Mappen js innehåller den Javascript fil som laddas in varje gång en ny vy visas. Javscript-koden används för att verifiera användarens input innan denna lagras i databasen.

I src mappen ligger den huvudsakliga applikationskoden och jag återkommer till denna nedan.

Styles-mappen innehåller en CSS-fil för att göra applikationen något trevligare att titta på.

Vendor mappen innehåller filer kopplade till Twig. Det är där som composer lagrar de filer eller moduler som installerats.

.gitignore filen används för att förhindra att ”vendor”-mappen samt ”deploy”-mappen versions hanteras.

Genom .htaccess filen styrs alla sidanrop om så att index.php filen körs oavsett sökväg.

Composer och composer.lock är filer osm används av Composer för att hantera insticksmoduler samt den autoloader som används för applikationen.

Dump.sql filen innehåller de kommandon som ”bygger” den ursprungliga databasen som används av applikationen.

Slutligen är index.php applikationens hjärta, den fil som hanterar hela applikationen.

En bild som visar skärmbild

Automatiskt genererad beskrivning

Src-mappen har flera mappar som innehåller de olika delarna av applikationen.

Controllers, Models och Views är ganska självförklarande.

Core-mappen innehåller filer som är gemensamma för applikationen. Där finns Config-klassen som används för att ta fram data ur config-filen. Där finns också en hjälp-klass, FilteredMap, som används för att ”tvätta” information som lämnas av användaren innan denna läggs in i databasen. Detta för att försöka minska risken för angrepp genom så kallad SQL-injection. I mappen finns också klasserna Request och Router som ansvarar för att styra användaren rätt i applikationen genom att analysera de angivna sökvägarna.

Mappen domain innehåller klasser som beskriver de objekt som applikationen använder sig av. Bilar, kunder och uthyrningar. Det är dessa objekt som skapas, modifieras, tas bort och lagras i databasen.

Exceptions innehåller klasser för att kunna avgöra vilka typer av fel som inträffar när applikationen används.

I mappen Utils slutligen så ligger den klass, DependencyInjector, som används för att skicka runt applikationsberoenden utan att behöva lista separata beroenden för de olika delarna av applikationen.