|  |
| --- |
|  |
| Software Architecture Document |
|  |
| CaféApp |

**SYSTEMINTEGRATÖR YHSIPI17/Systemarkitektur**

den 22 maj 2018

Billy Andersson, Malin Albinsson, Muhammad Assaf, Tommy Eriksson

Software Architecture Document

CaféApp

Innehåll

[2 Inledning 3](#_Toc515363786)

[2.1 Syfte 3](#_Toc515363787)

[2.2 Målgrupp 3](#_Toc515363788)

[2.3 Definitioner och förkortningar 3](#_Toc515363789)

[2.4 Referenser 3](#_Toc515363790)

[3 Arkitekturella mål och avgränsningar 3](#_Toc515363791)

[3.1 Övergripande mål 3](#_Toc515363792)

[3.2 Produkter och programspråk 3](#_Toc515363793)

[3.3 Systemets kringkomponenter 3](#_Toc515363794)

[3.4 Utvecklingsmiljö och utvecklingsverktyg 3](#_Toc515363795)

[4 Use case View (användarfallsvy) 4](#_Toc515363796)

[4.1 Översikt 4](#_Toc515363797)

[4.2 Aktörer 4](#_Toc515363798)

[4.2.1 Kund 4](#_Toc515363799)

[4.2.2 Personal 5](#_Toc515363800)

[4.2.3 Ägare 6](#_Toc515363801)

[4.2.4 Betalning 6](#_Toc515363802)

[4.3 Realisering av användarfall 7](#_Toc515363803)

[5 Logical View (Logisk vy) 7](#_Toc515363804)

[5.1 Översikt 7](#_Toc515363805)

[5.2 Interfacedefinitioner 7](#_Toc515363806)

[5.3 Tjänsteuppdelning 7](#_Toc515363807)

[5.4 Övergripande domänmodell 7](#_Toc515363808)

[6 Data View 7](#_Toc515363809)

[7 Deployment View (Driftsättningsvy) 8](#_Toc515363810)

[7.1 Lösningsöversikt 8](#_Toc515363811)

[7.2 Fysisk miljö 8](#_Toc515363812)

[7.3 Programvaror 8](#_Toc515363813)

[8 Icke-funktionella krav 9](#_Toc515363814)

[8.1 Svarstider och kapacitet 9](#_Toc515363815)

[8.2 Tillgänglighet 9](#_Toc515363816)

[8.3 Tillförlitlighet 9](#_Toc515363817)

[8.4 Förvaltningsbarhet 9](#_Toc515363818)

[9 Teknisk lösning 9](#_Toc515363819)

[9.1 Beskrivning 9](#_Toc515363820)

[9.2 Integration med omvärlden 9](#_Toc515363821)

[9.3 Felhantering 9](#_Toc515363822)

[9.3.1 Regler 9](#_Toc515363823)

[9.3.2 Dataformatering 9](#_Toc515363824)

[10 Säkerhet 10](#_Toc515363825)

[10.1 Säkerhetsklassificering av information 10](#_Toc515363826)

[10.2 Riskanalys 10](#_Toc515363827)

[10.3 Riskminimering 10](#_Toc515363828)

[10.4 Spårbarhet 10](#_Toc515363829)

# Inledning

## Syfte

## Målgrupp

Detta dokument är brett inriktat.

* Use case view riktar sig till samtliga, inklusive applikationens slutanvändare.
* Logical view har designers som sin primära målgrupp.
* Data view är främst avsedd för dataspecialister och databasadministratörer.
* Deployment view riktar sig till de som ansvarar för driften.

## Definitioner och förkortningar

## Referenser

# Arkitekturella mål och avgränsningar

## Övergripande mål

Huvudmålet med arkitekturen är att realisera beställarens krav på ett sådant sätt att systemet vid behov enkelt går att bygga ut.

## Produkter och programspråk

Följande produkter ingår i systemet:

|  |  |  |  |
| --- | --- | --- | --- |
| Produkttyp | Produkt | Version | Beskrivning |
| Driftskomponent |  |  |  |
| Databashanterare |  |  |  |
| Programspråk | Java | xxx | Suns Javaimplementation |

## Systemets kringkomponenter

## Utvecklingsmiljö och utvecklingsverktyg

Följande miljöer och verktyg kommer att användas för realisering, byggande och tester:

|  |  |
| --- | --- |
| Produkt | Ansvar |
| xxx | Programmeringsverktyg (IDE) |
| JUnit | Enhetstestning av kod |
| xxx | Byggstöd och rapporteringsverktyg |

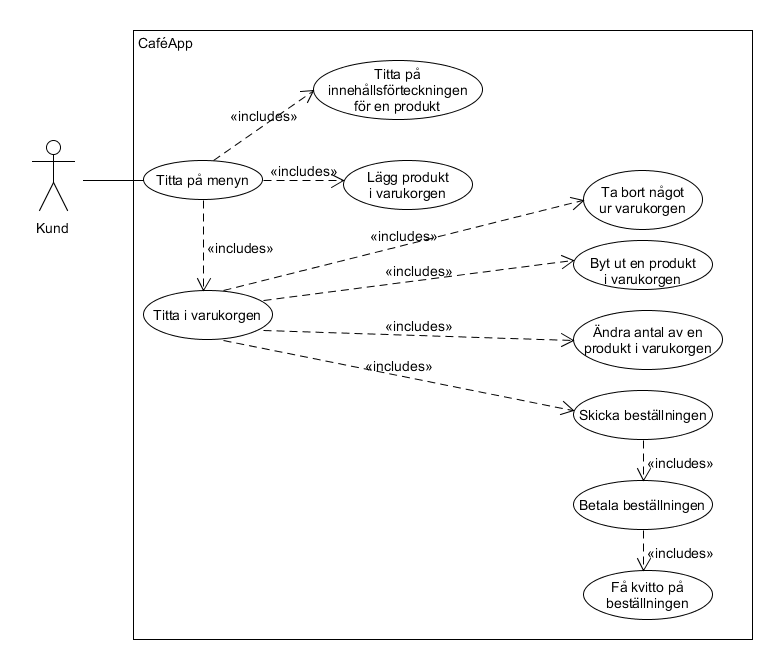
# Use case View (användarfallsvy)

## Översikt

## Aktörer

Systemet har fyra olika externa aktörer. Kund, Personal, Ägare och Betalning (exempelvis en bank).

### Kund

Kunden kan titta på menyn, granska innehållsförteckningen, välja produkter att lägga i varukorgen, titta och ändra i varukorgen och slutligen skicka beställningen, betala och få kvitto.

Figur 2 Use cases Kund



Figur 3 Några use cases för Kund i tabellform

### Personal

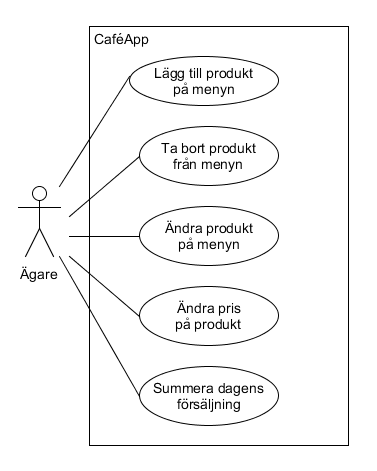
Personalen hanterar beställningen och ser till att kunden får sin mat och dryck. Eftersom kunderna också kan ställa frågor om produkterna direkt till personalen istället för att söka information i systemet ligger *Få fråga om produkt* utanför själva systemet, och då kan de vid behov ta fram information om produkten från systemet.



Figur 4 Use Cases Personal

### Ägare

Ägaren står för det administrativa, lägger till, tar bort och ändrar produkter på menyn och deras pris. Denne kan också ta ut rapporter ur systemet.



Figur 5 Use Cases Ägare

### Betalning

TODO – göra use case eller skriva något om varför inte

## Realisering av användarfall

TODO Sekvensdiagram för viktiga användarfall

# Logical View (Logisk vy)

## Översikt

## Interfacedefinitioner

## Tjänsteuppdelning

## Övergripande domänmodell

# Data View

Dataflödesdiagrammet level 0 visar dataflödet mellan externa entiteter och CaféApp-systemet. Vi har fyra olika externa entiteter/aktörer. Kund, Personal, Ägare och Betalning (exempelvis en bank).



Figur 1 DFD Level 0

Tittar vi på dataflödesdiagrammet för nästa nivå så ser vi att systemet inkluderar två databaser, en över de produkter caféet hanterar, och en över försäljningsdata. På den här nivån ser vi fyra processer, en för orderhantering, en för att hämta produktinformation, en för att administrera produkterna och en för att skapa rapporter.



Figur 6 DFD Level 1

# Deployment View (Driftsättningsvy)

## Lösningsöversikt

## Fysisk miljö

## Programvaror

# Icke-funktionella krav

## Svarstider och kapacitet

Från kundens perspektiv ska svarstiden normalt understiga 2 sekunder vid normalbelastning (20 samtidiga kunder). I övriga fall ska svarstiden vara mindre än 8 sekunder.

Systemet ska klara att hantera minst 50 samtidiga användare.

## Tillgänglighet

För kunden ska systemet vara tillgängligt minst under caféets öppettider. För personal och ägare ytterligare minst från två timmar innan caféet öppnar, samt fyra timmar efter stängning för att möjliggöra administrativa uppgifter.

Användargränssnittet ska vara intuitivt.

## Tillförlitlighet

Systemet bör gå att drifta med redundans.

## Förvaltningsbarhet

Det ska finnas separata miljöer för utveckling, test och produktion.

# Teknisk lösning

## Beskrivning

## Integration med omvärlden

## Felhantering

Javas Exceptions används för felhantering i applikationen.

### Regler

När ett fel inträffar ska problemet i första hand åtgärdas. Om felhanteringsrutinen inte kan åtgärda problemet ska det kastas vidare.

De fel som kastas ska i första hand vara egendefinierade, för att dölja implentationsdetaljer som inte bör framgå i det publika användargränssnittet. På så vis förbättras också systemets underhållsbarhet.

### Dataformatering

Datum skall formateras till ÅÅÅÅ-MM-DD format, t ex 2018-07-12.

Data som saknar värde, exempelvis en tom sträng, ska returneras som *null*.

# Säkerhet

## Säkerhetsklassificering av information

## Riskanalys

## Riskminimering

För att utveckla så säker programkod som möjligt har xxxxx använts för att förhindra SQL-injektioner.

## Spårbarhet