Sammanfattning

Google har nyligen lanserat programmeringsspråket Go. Det är företagets försök att skapa ett språk som är anpassat till de moderna sätten att programmera.

I rapporten utvärderas språket genom utveckling av ett webbuppslagsverk. Tiden för utvecklingen diskuteras och språket jämförs med programmeringsspråket Java inom områdena snabb utveckling, typhantering, webbprogrammering, trådhantering, grafiska användargränssnitt och utvecklingsverktyg.

Go har potential att bli ett populärt och användbart programmeringsspråk men språket lever inte upp till detta än.

Abstract

The Programming Language Go

Evaluation through Product Development

Google has recently launched the programming language Go. It is their attempt to create a language that is suitable for modern programming styles.

In this report the language is evaluated through product development of a web encyclopedia. The development time is discussed and the language is compared to the programming language Java in the areas of rapid development, type systems, web programming, concurrency, graphical user interfaces and development tools.

Go has the potential to become a popular and useful programming language but does not live up to its expectations yet.

Innehåll

Samarbete

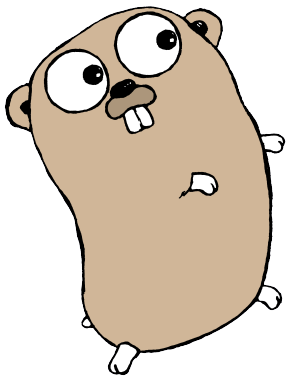
Detta är ett samarbete mellan David Falk och Klaus Nicosia. Båda författarna har deltagit i alla delar av rapporten (formgivning, skrivande samt korrekturläsning) och den tillhörande produktutvecklingen (planering, modellering, programmering samt testning).

KAPITEL 1

Inledning

1.1 Go och webbuppslagsverk

Google har nyligen lanserat ett nytt programmeringsspråk (*figur 1.1*) som har skapat så mycket hype så att det är svårt att veta om det är bra eller dåligt. Det är Googles försök att skapa ett språk som är anpassat till de nya sätten som används vid programmering. Ett exempel på detta är, trots att Go är kompilerat, att få med det bästa från både skriptspråk och kompilerade språk [1]. Dessutom är det tänkt att göra det lättare att utveckla program snabbt, bland annat med enkel syntax och känslan av dynamisk typhantering, utan att programmen blir långsamma att köra eller kompilera. Språket försöker göra det enklare att utveckla bra program för körning på flera kärnor. Vikt har också lagts på nätverksprogrammering, webbprogrammering och trådhantering.



Figur 1.1 *Programmeringsspråket Go:s maskot illustrerad av Renée French.*

För att utvärdera programmeringsspråket Go har vi programmerat en enkel wiki-webbsida i Go.

En wiki-webbsida är ett sökbart webbuppslagsverk med versionshantering där besökarna kan ändra och lägga till sidor via ett webbgränssnitt [2]. Wiki-webbsidan som utvecklas är inte helt komplett för att vi lägger inte in versionshantering. Fortsättningsvis används beteckningen webbuppslagsverk för vår wiki-webbsida.

Ett webbuppslagsverk är lämpligt för utvärderingen då det kräver att språket används inom många intressanta områden.

1.2 Syfte

Det är ovanligt att få möjligheten, och tid, att följa ett programmeringsspråk under de tidiga stegen av dess utveckling. Vi har fått denna möjligheten med Go vilket vi tycker har varit spännande. Ett intressant fortsättningsprojekt skulle vara att undersöka språket senare och notera vilka skillnader som har hänt med språket. Vi hoppas att få tillfälle till att göra det 2012.

Kompilerade språk eller skriptspråk? Valet styrs av de krav som det aktuella projektet ställer. Det är därför intressant att se om Go uppfyller sådana krav utan att tvinga fram ett val och samtidigt undersöka hur mycket av det bästa från båda världarna som Go kan erbjuda.

Som modernt språk är det också intressant att jämföra Go med äldre språk, inom områden som ofta förekommer i dag, såsom webbprogrammering, nätverksprogrammering och programmering för körning i flera trådar.

1.3 Frågeställning

Syftet med vår utvärdering är testa och analysera hur bra Go fungerar inom områden som snabb utveckling, typhantering, webbprogrammering, trådhantering, grafiska användargränssnitt och utvecklingsverktyg. Specifikt söker vi svar på dessa frågor:

1. Hur lång tid tar det för programmerare utan erfarenhet av Go att programmera ett enkelt webbuppslagsverk?

2. Hur utfaller en jämförelse mellan Go och Java inom de relevanta områdena?

Java väljs för jämförelsen i fråga två då det är ett populärt språk som vi har goda kunskaper inom. Vi är programmerare som inte hade erfarenhet av Go vid start av utvärderingen.

1.4 Metod

För att besvara första frågan ovan programmerade vi ett webbuppslagsverk och dokumenterade hur det gick. Vid programmeringen undersökte vi så mycket som

möjligt av de relevanta områdena så att vi kunde utvärdera dessa. Snabb utveckling, typhantering och webbprogrammering var oundvikliga områden vid programmering av webbuppslagsverket. Trådhantering var inte nödvändigt för webbuppslagsverket men vi utökade programmet med extra funktionalitet för att testa denna aspekt av Go. Speciellt har området snabb utveckling utvärderats genom att all programmering blev tidtagen. Vid jämförelsen har vi använt våra tidigare erfarenheter av Java inom de relevanta områdena och för tidsuppskattningar.

Vi delade upp projektet i tre delar: en inlärningsfas, en planeringsfas och en programmeringsfas.

I inlärningsfasen samlade vi information om Go från språkets hemsida. Då ett av våra syften var att testa språket inom snabb utveckling, fanns inte mycket tid att hitta många informationskällor, så bestämde vi oss för att endast använda språkets hemsida som inlärningsplats. Detta var inget problem eftersom sidan är innehållsrik; språket är väldokumenterat och det finns många självstudier (eng. tutorials).

Under planeringsfasen (*2.2 Planeringsfas*) formulerade vi grundläggande och avancerade krav som vårt program skulle uppfylla för att sedan starta programmeringsfasen.

Alla figurer och tabeller i denna rapport är skapade av rapportförfattarna. Det enda undantaget är *figur 1.1* som är licensierad under Creative Commons Attribution 3.0 och är illustrerad av Renée French [3].

KAPITEL 2

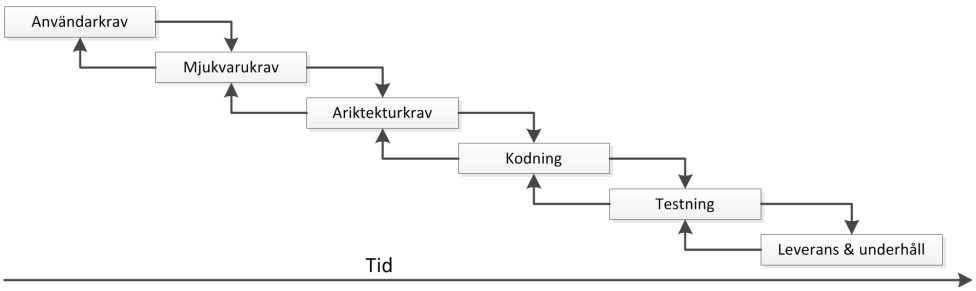
Utveckling av webbuppslagsverk

2.1 Vattenfallsmodellen

För att komma igång med projektet valde vi att använda en modell för mjukvaruutvecklingsprocesser. I en sådan process planeras och modelleras projektet innan man börjar programmera. Detta för att ha en tydlig struktur och för att programmeringen ska kunna utföras på bästa möjliga sätt.

Vattenfallsmodellen är en mjukvaruutvecklingsprocess som härstammar från mer allmänna ingenjörsprocesser [4]. Vi valde att dela upp utvecklingen enligt denna modell då det är den vi har störst erfarenhet av. Vattenfallsmodellen är uppdelad i sex faser och dessa är (*figur 2.1*):

- Användarkrav
- Mjukvarukrav
- Arkitekturkrav
- Kodning
- Testning
- Leverans och underhåll



Figur 2.1 *Vattenfallsmodellens sex olika faser.*

Användarkravs- och arkitekturkravsfasen utgör planeringsfasen i detta projekt. Eftersom projektet är så litet har vi valt att utelämna mjukvarukravsfasen. Kodnings- och testningsfasen utgör programmeringsfasen av projektet (*Kapitel 3: Resultat*). Det här projektet avslutas efter vattenfallsmodellens testningsfas så att leverans och underhållsfasen inte utförs, då vi inte har någon kund att leverera till.

2.2 Planeringsfas

Användarkravsfasen utgörs av att skriva ett *användarkravsdokument* (eng. *User Requirements Document, URD*) som beskriver vilka användningskrav vi har på den slutgiltiga produkten, se *Bilaga A: Användarkravsdokument*. *Tabell 2.1* nedan är en sammanfattning av krav formulerade i användarkravsdokumentet. I tabellen visas numreringen i dokumentet, identifierare och behov. Behov är en beskrivning av hur viktig kravet är att uppfyllas. I första hand uppfylls baskraven och om resurser tillåter så implementeras även avancerade krav.

Tabell 2.1 Sammanfattning av användarkrav i Bilaga A: Användarkravsdokument

Numrering	Identifierare	Behov
A.3.1.1	Sök på artikelnamn och visa artikel	Bas
A.3.1.2	Skapa artiklar	Bas
A.3.1.3	Ändra artiklar	Bas
A.3.1.4	Bas-alternativ vid skapa/ändra artiklar	Bas
A.3.1.5	Avancerade-alternativ vid skapa/ändra artiklar	Avancerad
A.3.1.6	Skapa artikel efter sökning	Bas
A.3.1.7	Slumpa artikel	Avancerad
A.3.1.8	Initiera om sidmallar	Avancerad
A.3.1.9	Ta bort artikel	Avancerad
A.3.1.10	Flera användare	Bas

Arkitekturskravsfasen utgörs av att skriva ett *arkitekturkravsdokument* (eng. *Architectural Design Document, ADD*) som är en modell av hur produkten ska fungera, se *Bilaga B: Arkitekturkravsdokument*.

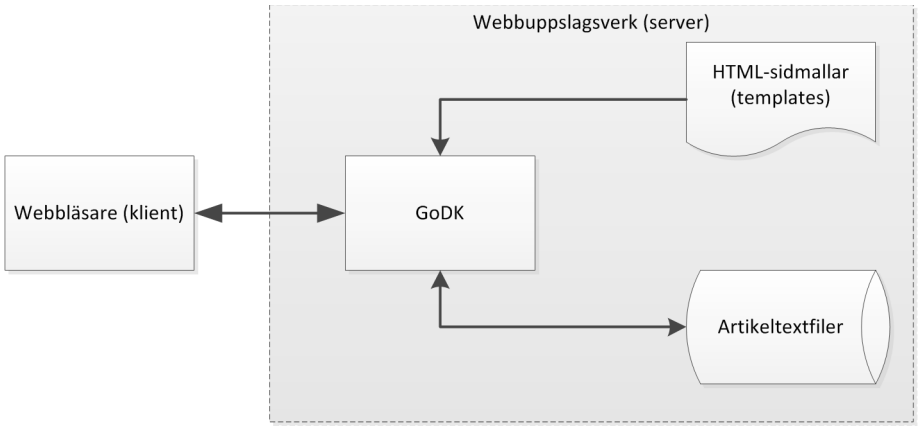
För att skriva användarkravs- och arkitekturskravsdokumenten har vi utgått från mallar av European Space Agency (ESA) [5]. Dessa mallar är redigerade av Karl Meinke för projekthandboken till mjukvarukonstruktionskursen vid KTH [6]. De redigerade mallarna har vi översatt till svenska för detta projekt.

Detta för att tydligt bestämma hur webbuppslagsverket ska se ut och fungera. Vanligtvis brukar det finnas ett tredje steg mellan användarkravsdokumentet och arkitekturkravsdokumentet som är ett *mjukvarukravsdokument* (eng. *Software Requirements Document, SRD*) men detta dokument anses onödigt då projektet med webuppslagsverket är så litet.

Under inlärningsfasen hittade vi en självstudie på Go:s hemsida som förklarar hur man skriver en webbapplikation i Go [7]. Självstudien visar ett exempel på hur ett enkelt webbuppslagsverk kan skrivas. Vi bestämde oss för att utgå från detta exempel och lägga till funktioner tills det blev användbart.

2.3 Webbuppslagsverkets delar

En webbläsare ansluter som klient till webbuppslagsverket som är server och består av tre komponenter (*figur 2.2*). GoDK står för **Go D**avid **K**laus och utgör huvudprogrammet som är skrivet i språket Go. HTML-sidmallar (templates) används för de färdiga sidorna i webbuppslagsverket. Alla artiklar sparas i separata textfiler.



Figur 2.2 *Övergripande komponentarkitektur.*

Webbuppslagsverket genererar, från HTML-sidmallarna och artikeltextfilerna, de färdiga sidorna:

- **Start**: Startsida där man kan söka efter artiklar.
- **Read**: Visar en specifik artikel om den existerar.
- **Edit**: Ändrar en specifik artikel eller skapar ny om den inte existerar.

■ **Notfound**: Denna sida visas om en sökning gjort på en artikel som inte finns. Möjlighet att skapa artikeln visas.

■ **Invalid**: Denna sida visas om artikelnamnet är otillåtet.

■ **Gemensamma delar**: Alla sidor i webbuppslagsverket utom Start har delar som är gemensamma. Här visas logotyp, länk till start och en sökruta. Detta innebär att det är möjligt att söka från andra sidor.

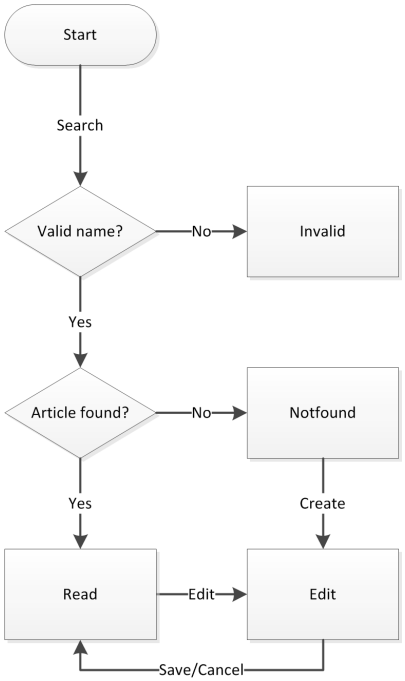
Användargränssnittsskisser presenteras i *Bilaga A: Användarkravsdokument*.

Flödesdiagrammet i *figur 2.3* visar användningen av webbuppslagsverket. Här kan man följa användarscenarion. Det framgår inte i figuren att det finns möjlighet att söka och ta sig till startsidan från alla sidor. Nedan presenteras några användarscenarion:

■ Från `Start` görs en sökning efter `G@`. `@` är ett ogiltigt tecken vilket innebär att villkoret `Valid name?` inte uppfylls och användaren blir omdirigerad till `Invalid`. Här ifrån kan användaren göra en ny sökning eller gå tillbaka till `Start.`

■ Från `Start` görs en sökning efter `Go`. Detta är ett giltigt namn och villkoret `Valid name?` uppfylls så användaren blir omdirigerad till villkoret `Article Found?`. I detta scenario finns inte artikeln `Go` så användaren omdirigeras till `Notfound`. Här väljer användaren att skapa artikeln och omdirigeras till `Edit`. I detta steg redigeras artikeln tills användaren är nöjd. När artikeln sparas omdirigeras användaren till `Read` och får se den nya `Go`-artikeln.

■ Från `Start` görs en sökning efter `Go`. Detta är ett giltigt namn och villkoret `Valid name?` uppfylls så användaren blir omdirigerad till villkoret `Article Found?`. Nu finns artikeln `Go` så användaren omdirigeras till `Read` och artikeln visas. Användaren som vill ändra artikeln klickar på edit-knappen och omdirigeras till `Edit`. Här redigerar användaren artikeln. När artikeln sparas omdirigeras användaren tillbaka till `Read` och får se den redigerade `Go`-artikeln.



Figur 2.3 *Flödesschema för webbuppslagsverket.*

KAPITEL 3

Resultat

3.1 Uppfyllda krav

I *tabell 2.1* ovan presenterades vilka krav på webbuppslagsverket vi hade för avsikt att implementera. Här följer en genomgång av vilka krav vi har lyckats att uppfylla. En sammanfattning visas i *tabell 3.1*.

- **Sök på artikelnamn och visa artikel:** Det är möjligt att söka i webbuppslagsverket efter artikelnamnet.

- **Skapa artiklar:** En användare kan skapa artiklar som sedan finns tillgängligt i webbuppslagsverket.

- **Ändra artiklar:** En artikel kan ändras och sparas.

- **Bas-alternativ vid skapa/ändra artiklar:** Vid ändring av artiklar kan formateringar göras. De enklare formateringar som kan utföras är: fet stil, kursiv stil, understruken stil, tre olika storlekar på text och ny rad.

- **Avancerade-alternativ vid skapa/ändra artiklar:** De avancerade formateringar som kan utföras är: länk till annan artikel, länk till annan webbsida och lägga in bilder.

- **Skapa artikel efter sökning:** Om en artikel inte finns efter en sökning har användaren möjlighet att skapa artikeln.

- **Slumpa artikel:** Det är inte möjligt att slumpa artiklar, kravet är inte uppfyllt. Vi har inte försökt implementera denna funktion på grund av tidsbrist.

- **Initiera om sidmallar:** Från programmet kan administratören initiera om sidmallar utan att starta om programmet, så att formgivningen för webbsidan kan ändras under körning.

■ **Ta bort artikel:** Från programmet kan administratören ta bort artiklar under körning.

■ **Flera användare:** Flera användare kan använda webbuppslagsverket samtidigt. Däremot, om en artikel redigeras samtidigt av flera användare så är det endast den version som sparas sist som blir tillgänglig i webbuppslagsverket.

Tabell 3.1 Sammanfattning av användarkrav i Bilaga A: Användarkravsdokument

Numrering	Identifierare	Behov	Uppfyllt
A.3.1.1	Sök på artikelnamn och visa artikel	Bas	✓
A.3.1.2	Skapa artiklar	Bas	✓
A.3.1.3	Ändra artiklar	Bas	✓
A.3.1.4	Bas-alternativ vid skapa/ändra artiklar	Bas	✓
A.3.1.5	Avancerade-alternativ vid skapa/ändra artiklar	Avancerad	✓
A.3.1.6	Skapa artikel efter sökning	Bas	✓
A.3.1.7	Slumpa artikel	Avancerad	✗
A.3.1.8	Initiera om sidmallar	Avancerad	✓
A.3.1.9	Ta bort artikel	Avancerad	✓
A.3.1.10	Flera användare	Bas	✓

3.2 Tidsdata

Under arkitekturkravsfasen görs en tidsplanering för programmets komponenter. Då oerfarenhet inom språket leder till osäkerhet har vi kompenserat detta genom att lägga till tid för inlärning i planen. Dessutom har vi uppskattat tider för vad samma projekt skulle ta i programmeringsspråket Java. Dessa kan jämföras med den verkliga tiden som togs vid programmeringen. Vid samarbete dubblerades tiden då det gick åt dubbla antal mantimmar. Se *tabell 3.2*.

Projektets programmeringsfas utgörs av vattenfallsmodellens kodning- och testningsfas och tog 48,5 timmar. GoDK är den enda komponenten som involverar ren Go-programmering där ungefär 33 timmar lagts. Det är inte ovanligt att webbprogrammering involverar HTML, CSS och JavaScript. Go är inget undantag och cirka 15,5 timmar har tillbringats på detta i HTML-sidmallar: Gemensamma delar, Start, Read, Edit, Notfound och Invalid.

Tabell 3.2 Tidsdata

Komponent	Planerad tid	Uppskattad tid i Java	Verklig tid
GoDK	25 h	35 h	33 h
Gemensamma delar	10 h	15 h	7 h
Start	2 h	2 h	0,5 h
Read	5 h	5 h	1 h
Edit	10 h	10 h	6 h
Notfound	2 h	2 h	0,5 h
Invalid	2 h	2 h	0,5 h
Totalt	**56 h**	**71 h**	**48,5 h**

Relevant för tidsåtgången för oss som nybörjarprogrammerare inom Go är också tiden som tillbringades i inlärningsfasen och planeringsfasen (*2.2 Planeringsfas*). Sex timmar tillbringades på självstudien i inlärningsfasen. Planeringsfasen tog 16 timmar, där cirka åtta timmar tillbringades på vardera dokument (*Bilaga A: Användarkravsdokument* och *Bilaga B: Arkitekturkravsdokument*). Detta har givetvis minskat tiden för programmeringen.

För att utvärdera språket utöver det som behövdes för att uppfylla användarkraven ovan (*3.1 Uppfyllda krav*) implementerades andra funktioner. Exempelvis implementerades omdirigering (redirect) mellan artiklar. Omdirigering används när flera sökningar ska ta användaren till samma artikel. Tiden för dessa funktioner har räknats in i tiden för GoDK.

3.3 Skärmdumpar

Nedan visas skärmdumpar på det färdiga sidorna i programmet. Dessa har en gemensam del som gör det möjligt att söka artiklar och navigera till start-sidan från alla sidor.

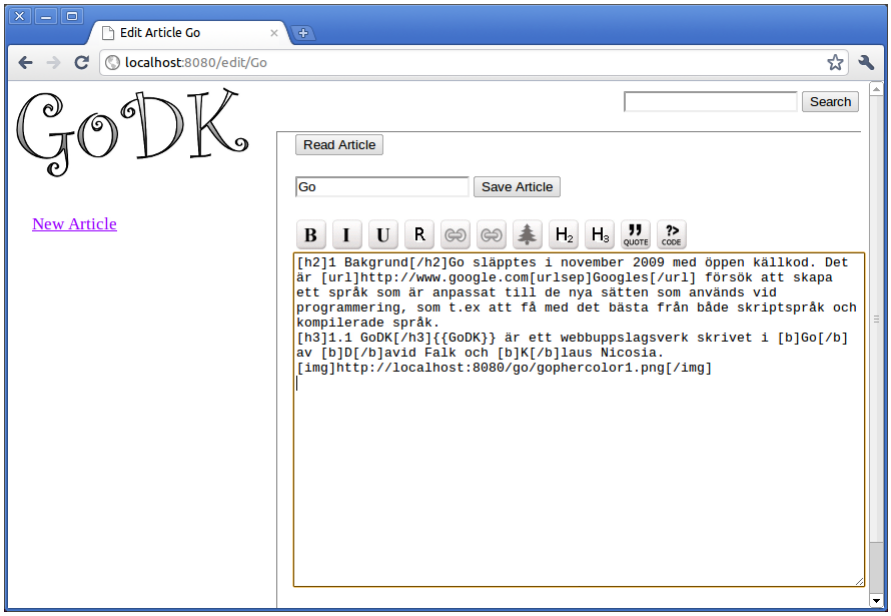
Start-sidan ger användaren möjlighet att söka efter artiklar i webbuppslagsverket (*figur 3.1*).

Figur 3.1 *Skärmdump av den färdiga Start-sidan.*

Read-sidan ger användaren möjlighet att läsa artiklar i webbuppslagsverket. Här finns en länk till edit-sidan där det finns möjlighet att redigera artikeln (*figur 3.2*).

Figur 3.2 *Skärmdump av den färdiga Read-sidan.*

Edit-sidan ger användaren möjlighet att ändra eller skapa artiklar. För att hjälpa användaren med detta finns formateringsalternativ som knappar ovanför textrutan. Det finns även knappar för att spara artikeln och för att läsa artikel utan att spara ändringarna (*figur 3.3*).

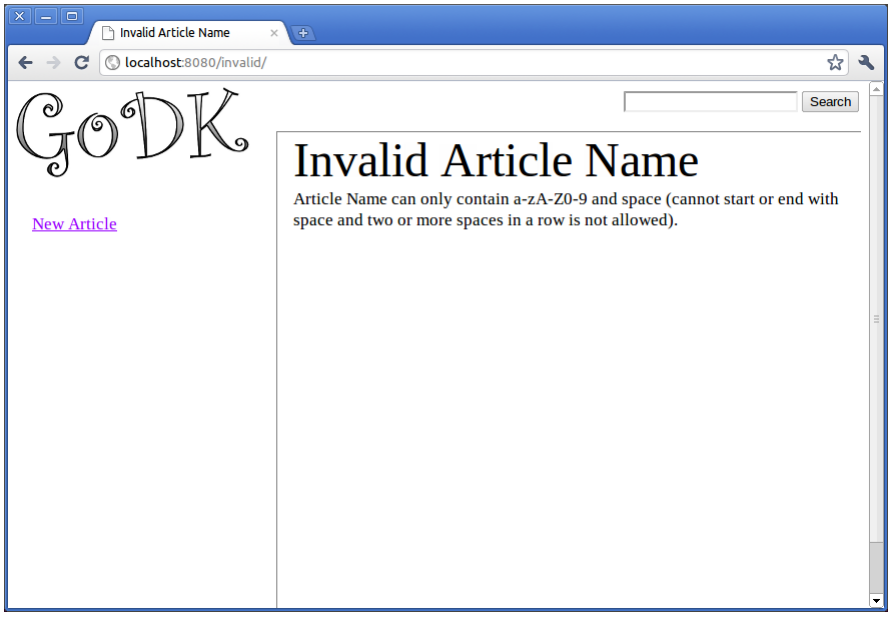


Figur 3.3 *Skärmdump av den färdiga Edit-sidan.*

Notfound-sidan visas när en artikel som användaren har sökt efter inte finns. Då kan användaren välja att skapa artikeln (*figur 3.4*).

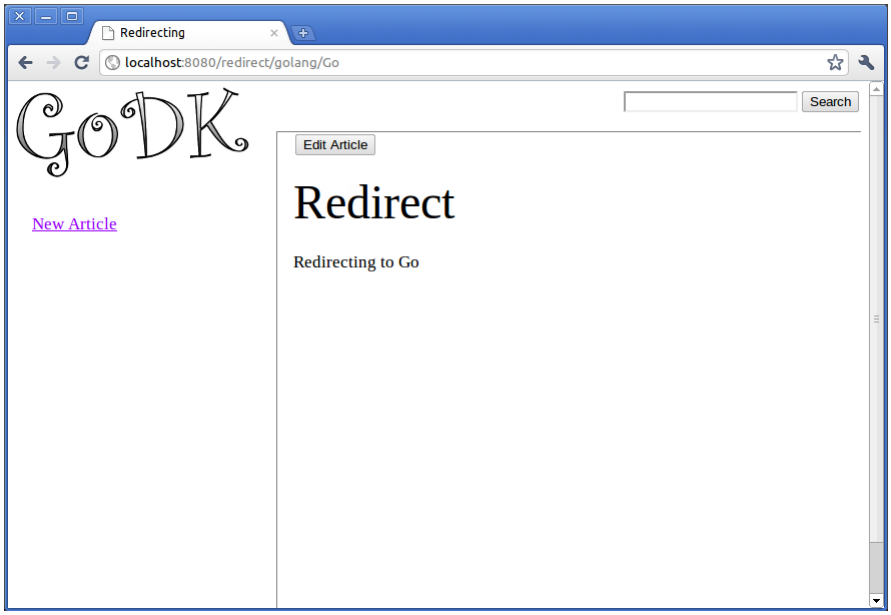
Figur 3.4 *Skärmdump av den färdiga Notfound-sidan.*

Invalid-sidan visas när man sökt efter en artikel med ogiltigt namn. På sidan visas vilka tecken som är tillåtna (*figur 3.5*).



Figur 3.5 *Skärmdump av den färdiga Invalid-sidan.*

Redirect-sidan är en sida som visas i tre sekunder för att sedan omdirigera användaren till en annan artikel. Här finns en länk till edit-sidan där användaren kan ta bort eller ändra omdirigeringen (*figur 3.6*).



Figur 3.6 *Skärmdump av Redirect-sidan.*

KAPITEL 4

Diskussion

4.1 Tidtagning

Första frågan i *1.3 Frågeställning* var:

> Hur lång tid tar det för en programmerare utan erfarenhet av Go att programmera ett enkelt webbuppslagsverk?

Det tog oss 48,5 timmar varav 33 timmar var Go-programmering, i GoDK-komponenten, att programmera webbuppslagsverket. Hela 15,5 timmar tillbringades, i övriga komponenter, på HTML, CSS och JavaScript. Vi blev besvikna på att så mycket tid behövde ägnas åt detta när man programmerar i Go.

Ett modernt språk, med fokusering på bland annat webbprogrammering, borde kunna utföra formgivningen direkt som en del av koden. På grund av detta beräknade vi den planerade tiden under antagandet att alla komponenter skulle innehålla Go-programmering (*tabell 3.2)*. Detta var inte möjligt och all Go-kod fick läggas i GoDK, vilket i sin tur ledde till att den komponenten tog längre tid än beräknat och övriga komponenter tog kortare tid. Trots att GoDK-komponenten tog längre tid att utveckla blev den verkliga tiden, för hela webbuppslagsverket, lägre än den planerade tiden.

Under planeringen uppskattade vi tiden för projektet om det skulle programmeras i Java. Verkliga tiden för webbuppslagsverket i Go blev betydligt lägre än den uppskattade tiden för Java.

4.2 Jämförelse med Java

Andra frågan i *1.3 Frågeställning* var:

> Hur utfaller en jämförelse mellan Go och Java inom de relevanta områdena?

Dessa områden är: snabb utveckling, typhantering, webbprogrammering, trådhantering, grafiska användargränssnitt och utvecklingsverktyg.

Snabb utveckling

Området snabb utveckling domineras framför allt av skriptspråk och Go påstås ha samma fördelar som dessa. Det stämmer inte än, då syntaxen inte är lika enkel som den brukar vara i skriptspråk. Jämfört med Java är i princip den enda förenklingen att rader inte behöver avslutas med semikolon. Funktions- och variabeldeklarationer i Go är skrivna i omvänd ordning mot Java. Nedan följer några exempel på skillnader i syntax mellan Go och Java.

En variabel deklareras i Go:

```
var i int                                                    (1)
```

En variabel deklareras i Java:

```
int i;                                                       (2)
```

Variabeldeklarationer är alltså längre i Go (1) än i Java (2). Notera att typen (`int`) står till höger om variabelnamnet (`i`) i Go-kodexemplet (1).

En variabel definieras i Go:

```
j = getInt()                                                 (3)
```

En variabel definieras i Java:

```
j = getInt();                                                (4)
```

Variabeldefinitioner är sånär som på ett semikolon identiska i Go (3) och i Java (4).

En variabel kan samtidigt deklareras och definieras på tre sätt i Go:

```
var k int = getInt(),                                        (5)
var k = getInt() eller                                       (6)
k := getInt()                                                (7)
```

En variabel deklareras och definieras i Java:

```
int k = getInt();
```
(8)

Två av tre sätt att kombinera definition med deklaration av en variabel är alltså längre i Go som syns i kodexempel (5) och (6) jämfört med Java (8). Det tredje sättet i Go (7) är kortare och utnyttjar en del av typhanteringen, se *Typhantering* nedan.

Typhantering

Go är statisk typat men har element som påminner om dynamisk typning, för att likna skriptspråk. Om typen går att få reda på genom definitionen behöver den inte anges explicit, detta är vad som sker i kodexempel (6) och (7) ovan (*Snabb utveckling*). Denna kontroll sker under kompilering och inte under körning.

Go är hårdvarunära likt C, med exempelvis pekare, som bidrar till att typningen är stark. Java är som Go statiskt och starkt typat men gör detta mer konsekvent. Att inte typen skrivs explicit i Go är ingenting som nämnvärt förenklar programmeringen.

Webbprogrammering

Som nämnts tidigare (*4.1 Tidtagning*) hade vi önskat att mer av formgivningen hade kunnat utföras direkt med Go. Språket är byggt från grunden med avseende på att vara bra på webbprogrammering och trots att HTML-, CSS- och JavaScript-kunskaper krävs har det fungerat bra; webbprogrammering är mycket enklare i Go än i Java. För att få igång en HTTP-server som lyssnar på en port krävs bara en rad Go-programkod:

```
http.ListenAndServe(":" + port, nil)
```
(9)

För att koppla varje URI, exempelvis `'/'`, med en Go funktion som visar en sida, exempelvis `example.html`, behövdes ytterligare en rad programkod:

```
http.HandleFunc("/", exampleHandler)
```
(10)

För att generera sidan används funktionen `template.ParseFile` och metoden `execute` i paketet `template`. I exemplet substitueras alla förekomster av `{Head}` och `{Body}` med strängarna `head` respektive `body` i `example.html`.

```
func exampleHandler(w http.ResponseWriter,
                        r *http.Request) {
   head := "This is my head"
   body := "This is my body"
   s := &MyStruct{Head: head, Body: body}
   t, error := template.ParseFile("example.html", nil)
   if error == nil {
      t.Execute(w, s)
   }
}
```
(11)

Trådhantering

I jämförelse med Java är det enkelt att använda trådar och det kräver inte mycket erfarenhet. Go använder *goroutines* som kör funktioner i olika trådar om det behövs [8]. När en funktion som körs i en goroutine blockeras, till exempel av ett systemanrop, flyttas andra goroutines över till andra trådar som fortsätter köra parallellt med den förstnämnda tråden. Det enda som programmeraren behöver skriva är `go func()`, där `func(){ ... }` är funktionen som ska ligga i den nyskapade goroutinen.

I Java skapas nya trådar genom att instansiera ett objekt av en klass som implementerar gränssnittet (eng. *interface*), `Runnable`. `Runnable` har metoden `start()` som används för att starta tråden.

Grafiska användargränssnitt

Det är möjligt att göra avancerade grafiska användargränssnitt i Java. På grund av Javas plattformsoberoende och avancerade funktionalitet är detta ganska komplicerat och tidskrävande i jämförelse med vissa moderna språk.

Go som är fokuserat på webbprogrammering har i nuläget inget färdigt i sitt *API* för utveckling av grafiska användargränssnitt. Eftersom källkoden för Go:s API är öppen är det inte omöjligt att någon annan utvecklar denna funktionalitet. Det är dock inte troligt att Go:s utvecklare själva lägger ned tid på detta då tanken är att framförallt använda webbgränssnitt.

Det har börjat komma en del *språkbindningar* (eng. l*anguage bindings*) till API:er för grafiska användargränssnitt i andra språk [9]. Ett exempel på detta är *go-gtk* som möjliggör användning av populära *GTK+* (*GIMP Toolkit*) som är skrivet i språket *C* [10].

Utvecklingsverktyg

Det finns ett stort utbud på utvecklingsverktyg för Java vilket inte är fallet för det unga språket Go. Detta är en av anledningarna till att det är enklare att utföra stora programmeringsprojekt i Java. De flesta som skriver program i Go använder en vanlig textredigerare men det börjar dyka upp andra utvecklingsverktyg. Ett exempel är *Goclipse* som är ett *insticksprogram* (eng. *plug-in*) till utvecklingsverktyget *Eclipse* [11].

Övrigt

Den största skillnaden mellan Go och Java är naturligtvis att Java är objektorienterat men inte Go, även om det påstås att det går att skriva i Go med en objektorienterad stil [12]. Go har funktioner och vad som i Go kallas metoder. Dessa metoder är dock inte vad som vanligen menas inom objektorienterad programmering utan egentligen funktioner med ett argument (`struct` som ska likna objektet som anropar metoden) som har flyttats. Nedan visas skillnaderna på dessa i Go.

En vanlig funktion i Go:

```
func myFunction(s *MyStruct, a int) { ... }
```
(12)

Den anropas genom:

```
myFunction(*s, a)
```
(13)

En metod i Go, notera att argumentet `s* MyStruct` har flyttats framför metodnamnet:

```
func (s *MyStruct) myMethod(a int) { ... }
```
(14)

Den anropas genom:

```
s.myMethod(a)
```
(15)

Dessa exempel gör samma sak men anropas på olika sätt. Lösningen känns onödig och kanske mer förvirrande än hjälpsam när andra fördelar med objektorientering inte finns. Det vore bättre om språket var helt objektorienterat eller inte alls.

Stränghanteringen i Go var mycket komplicerad jämfört med den i Java. Det är troligt att detta beror på att språket fortfarande är under utveckling och all funktionalitet inte har hunnit implementeras. Under utvecklingen av webbuppslagsverket behövde vi göra första tecknet, och endast det tecknet, i en sträng till stor bokstav. Med Go:s aktuella API behövde använda *ASCII*-koden för tecken i strängen för att utföra detta. För att utföra detta på strängen `word` i Go var vi tvungna att skriva följande omständliga kod:

```
if (word[0] >= 97) {
   firstLower := string(word[0])
   firstUpper := string(word[0] - 32)
   word = strings.Replace(word, firstLower, firstUpper, 1)
}
```
(16)

I Java hade man kunnat utföra samma sak på strängen `word` genom att skriva:

```
word = word.substring(0,1).toUpperCase()
           + word.substring(1);
```
(17)

Java är ett programmeringsspråk med många bra funktioner för att läsa data från tangentbord. Go kan inte alls konkurrera med den funktionaliteten då dess inläsningsfunktioner än så länge är omoderna och komplicerade att använda.

Med Go blev en del komplicerade saker vid programmeringen enkla, till exempel trådhantering. Däremot blev en del enklare saker komplicerade, exempelvis stränghantering. Det märks tydligt att språket är nytt och att dess API inte är så omfattande ännu.

4.3 Felkällor

Tidsplanering innebär alltid osäkerhet och den planerade tiden för webbuppslagsverket är inget undantag. Eftersom vi inte hade programmerat i Go tidigare blev tidsplanen ännu mer osäker.

Vi var tyvärr oerfarna i webbprogrammering med Java vilket bidrog till felmarginaler vid tidsuppskattningen för samma projekt i Java.

Webbuppslagsverkets formgivning gjordes i HTML, CSS och JavaScript. Då vi inte var erfarna i denna typ av programmering kan det ha bidragit till längre utvecklingstider för de delarna av webbuppslagsverket.

4.4 Slutsats

Det var förvånansvärt enkelt att programmera ett webbuppslagsverk i Go för oss som var nybörjare i språket. Detta för att webbprogrammering och trådhantering var väldigt enkelt i Go till skillnad från Java. Problematiskt i Go var framför allt stränghantering och inläsning från tangentbord.

Det känns omständligt att ha en syntax som skiljer sig så mycket från populära programmeringsspråk. De omvända deklarationerna i Go är det som sticker ut och tar onödig tid att vänja sig vid.

Vi har inte haft möjlighet att testa webbservern inom aspekter så som säkerhet och hög belastning. Frågan är om den inbyggda webbserverfunktionaliteten i Go kan mäta sig med etablerade webbservrar så som *Apache* och *Microsoft IIS*. Om inte är det möjligt att det vore bättre att separera programmering på serversidan, alltså hantering av dynamiska sidor med mera, från webbservern.

Go har potential att bli ett populärt och användbart programmeringsspråk men det är ett språk under utveckling som saknar mycket funktionalitet så det är svårt att göra en rättvis bedömning i ett sådant tidigt skede.

KAPITEL 5

Referenser

5.1 Elektroniska källor

[1] Google (2011). The Go Programming Language, *The Go Programming Language*. [www] <http://golang.org/> 2011-02-20.

[2] Wikimedia Foundation (2011). Wikipedia, *Wiki*. [www] <http://sv.wikipedia.org/wiki/Wiki> 2011-02-20.

[3] Google (2011). The Go Programming Language, *FAQ*. [www] <http://golang.org/doc/go_faq.html#Whats_the_origin_of_the_mascot> 2011-04-11.

[5] European Space Agency (2011). ESA, *Software engineering & standardisation*. [www] <http://www.esa.int/TEC/Software_engineering_and_standardisation/TECBUCUXBQE_0.html> 2011-04-12.

[6] Meinke, Karl (2011). KTH, DD1365 Mjukvarukonstruktion, *Student Handbook*. [www] <http://www.nada.kth.se/~karlm/mvk/Project_Handbook.pdf> 2011-04-12.

[7] Google (2011). The Go Programming Language, *Codelab: Writing Web Applications*. [www] <http://golang.org/doc/codelab/wiki/> 2011-04-12.

[8] Google (2011). The Go Programming Language, *FAQ*. [www] <http://golang.org/doc/go_faq.html#goroutines> 2011-05-29.

[9] Cat-v (2011). Cat-v.ORG, *Go Bindings for Various External APIs*. [www] <http://go-lang.cat-v.org/library-bindings> 2011-05-29.

[10] The GTK+ Team (2011). GTK+, *About*. [www] <http://www.gtk.org/> 2011-05-29.

[11] Goclipse (2011). goclipse, *Eclipse-based IDE for Google Go*. [www] <http://code.google.com/p/goclipse/> 2011-05-30.

[12] Google (2011). The Go Programming Language, *FAQ*. [www] <http://golang.org/doc/go_faq.html#Is_Go_an_object-oriented_language> 2011-04-14.

5.2 Tryckta källor

[4] Sommerville, Ian (2007). *Software Engineering*. 8. uppl. Addison-Wesley Publishers Limited. ISBN: 0-321-31379-8.

Bilagor

A Användarkravsdokument

1 Introduktion

1.1 Syfte

Dokumentets syfte är att beskriva användarkrav för det webbuppslagsverk som ska utvecklas.

1.2 Mjukvarans omfattning

Skapa ett sökbart webbuppslagsverk där besökarna kan söka efter, läsa, ändra och lägga till artiklar.

1.3 Definitioner, akronymer och förkortningar

Go - Googles programmeringsspråk som kommer att användas i detta projekt [1].

GoDK - Webbuppslagsverket som utvecklas: **Go D**avid **K**laus Wiki.

1.4 Referenser

[1] Google (2011). The Go Programming Language, *The Go Programming Language*. [www] <http://golang.org/> 2011-02-20.

1.5 Överblick av dokumentet

2 Allmän beskrivning

2.1 Produktperspektiv

Endast för testning av programmeringsspråket Go.

2.2 Allmänna krav

Webbsidan kommer att bestå av 6 delar:

1. **Gemensamma delar**: Delar av webbuppslagsverket som är gemensamma för alla sidor utom Start. Här visas logotyp, länk till start och en sökruta. Detta innebär att det är möjligt att söka från andra sidor.

2. **Start**: Startsida där man kan söka efter artiklar, se *figur A.1*.

Figur A.1 *Skiss för Start-sidans användargränssnitt.*

3. **Read**: Visar en specifik artikel om den existerar, se *figur A.2*.

Figur A.2 *Skiss för Read-sidans användargränssnitt.*

4. **Edit**: Ändrar en specifik artikel eller skapar ny om den inte existerar, se *figur A.3*.

Figur A.3 *Skiss för Edit-sidans användargränssnitt.*

5. **Notfound**: Denna sida visas om en sökning gjort på en artikeln som inte finns. Möjlighet att skapa artikeln visas, se *figur A.4*.

Figur A.4 *Skiss för Notfound-sidans användargränssnitt.*

6. **Invalid**: Denna sida visas om artikelnamnet är otillåtet, se *figur A.5*.

Figur A.5 *Skiss för Invalid-sidans användargränssnitt.*

2.3 Allmänna restriktioner

Tidsbegränsningar

Eftersom projektet ska utföras snabbt kommer många avancerade funktioner inte att ha hög prioritet och kan då bli utelämnade.

Hårdvarubegränsningar

Webbuppslagsverket kommer utvecklas och testas på persondatorer som inte har samma prestanda som en server i praktiken skulle ha haft. Programmet kan därför inte testas i verkliga scenarion t.ex. hög internettrafik och stort antal artiklar.

2.4 Användarprofil

Webbuppslagsverket kommer att ha tre typer av användare:

- **Administratör**: Startar och avslutar programmet, har möjlighet att ta bort artiklar. Finns på serversidan av webbuppslagsverket.
- **Artikelläsare**: Söker efter och läser artiklar. Finns på klientsidan av webbuppslagsverket.
- **Artikelskapare**: Skapar eller ändrar artiklar. Finns på klientsidan av webbuppslagsverket.

2.5 Antaganden och beroenden

- Webbläsare som är klient vid anslutning till servern som utgör webbuppslagsverket.
- Om server och klient körs på olika datorer krävs internetförbindelse mellan dessa.
- Hårdvaran tillåter skrivning till filsystem.

2.6 Operationsmiljö

HTTP används för kommunikation mellan server (webbuppslagsverk) och klient (webbläsare).

3 Specifika krav

3.1 Funktionskrav

3.1.1 Sökning med artikelnamn

Identifierare	Sök på artikelnamn och visa artikel
Kravbeskrivning	Sökning efter artiklar i webbuppslagsverket med hjälp av artikelnamn ska visa artikeln.
Motivering	Användarna ska kunna söka och visa existerande artiklar.
Behov	Bas
Prioritet	Hög
Stabilitet	Stabil
Källa	David Falk och Klaus Nicosia
Verifierbarhet	Prova att söka efter artiklar och se att rätt artikel visas.

3.1.2 Skapa artiklar

Identifierare	Skapa artiklar
Kravbeskrivning	Man ska kunna skapa artiklar.
Motivering	Användare som vill ha fler artiklar i wikin ska kunna lägga till artiklar.
Behov	Bas
Prioritet	Hög
Stabilitet	Stabil
Källa	David Falk och Klaus Nicosia
Verifierbarhet	Prova att lägga till artiklar och se om de hamnar tillgängliga i webbuppslagsverket.

3.1.3 Ändra artiklar

Identifierare	Ändra artiklar
Kravbeskrivning	Man ska kunna ändra existerande artiklar.
Motivering	Om en användare hittar ett fel i någon artikel ska denna användare kunna rätta till detta felet.
Behov	Bas
Prioritet	Hög
Stabilitet	Stabil
Källa	David Falk och Klaus Nicosia
Verifierbarhet	Prova att ändra en existerande artikel och se om ändringen sparas och är tillgänglig i webbuppslagsverket.

3.1.4 Bas-alternativ vid skapa/ändra artiklar

Identifierare	Bas-alternativ vid skapa/ändra artiklar
Kravbeskrivning	Enkel formatering av text ska kunna göras när man ändrar eller lägger till artiklar. Det som ska kunna göras är: ■ Fet stil ■ Kursiv stil ■ Understruken stil ■ Tre olika storlekar på text ■ Användande av ny rad
Motivering	En artikel ska vara trevlig att läsa och inte bara vara en klump med text.
Behov	Bas
Prioritet	Hög
Stabilitet	Stabil
Källa	David Falk och Klaus Nicosia
Verifierbarhet	Skriv en artikel med simpel formatering och se om artikeln har simpel formatering när den visas.

3.1.5 Avancerade-alternativ vid skapa/ändra artiklar

Identifierare	Avancerade-alternativ vid skapa/ändra artiklar
Kravbeskrivning	Avancerad formatering av text ska kunna göras när man ändrar eller lägger till artiklar. Det som ska kunna göras är: ■ Länka till annan artikel ■ Länka till annan webbsida ■ Lägga in bilder i artikeln
Motivering	En artikel ska kunna ge läsaren extra information till andra artiklar och kunna visa bild på intressanta saker.
Behov	Avancerad
Prioritet	Låg
Stabilitet	Stabil
Källa	David Falk och Klaus Nicosia
Verifierbarhet	Skriv en artikel med avancerad formatering och se om artikeln har avancerad formatering när den visas.

3.1.6 Skapa artikel efter sökning

Identifierare	Skapa artikel efter sökning
Kravbeskrivning	Efter man sökt efter en artikel som inte fanns ska det finnas ett alternativ att skapa denna artikel.
Motivering	Man ska snabbt kunna skapa artiklar som man märker inte finns.
Behov	Bas
Prioritet	Hög
Stabilitet	Stabil
Källa	David Falk och Klaus Nicosia
Verifierbarhet	Testa att söka efter en artikel som inte finns och se om man får alternativ att skapa artikeln.

3.1.7 Slumpa artikel

Identifierare	Slumpa artikel
Kravbeskrivning	Användare ska kunna trycka på en slumpknapp och få fram en slumpad artikel från webbuppslagsverket.
Motivering	Användare ska kunna börja läsa från webbuppslagsverket direkt genom att slumpa en artikel.
Behov	Avancerad
Prioritet	Låg
Stabilitet	Stabil
Källa	David Falk och Klaus Nicosia
Verifierbarhet	Testa att klicka på knappen som slumpar artikel och se om en artikel slumpas.

3.1.8 Initiera om sidmallar

Identifierare	Initiera om sidmallar
Kravbeskrivning	Ladda om sidmallar (templates) till programmet under körning.
Motivering	Man ska kunna ändra hur sidorna ser ut utan att ta ner webbuppslagsverket om man vill det.
Behov	Avancerad
Prioritet	Låg
Stabilitet	Stabil
Källa	David Falk och Klaus Nicosia
Verifierbarhet	Testa att initiera om sidmallarna under körning.

3.1.9 Ta bort artikel

Identifierare	Ta bort artikel
Kravbeskrivning	Administratören ska kunna ta bort artiklar från webbuppslagsverket via programmet.
Motivering	Ibland innehåller sidor saker som inte administratören vill ha med och då ska dessa sidor kunna tas bort.
Behov	Avancerad
Prioritet	Låg
Stabilitet	Stabil
Källa	David Falk och Klaus Nicosia
Verifierbarhet	Testa att ta bort artiklar via programmet.

3.1.10 Flera användare

Identifierare	Flera användare
Kravbeskrivning	Flera användare ska kunna läsa artiklar från webbuppslagsverket samtidigt.
Motivering	Flera användare besöker sidan samtidigt.
Behov	Bas
Prioritet	Hög
Stabilitet	Stabil
Källa	David Falk och Klaus Nicosia
Verifierbarhet	Testa att flera användare besöker webbuppslagsverket samtidigt.

3.2 Restriktionskrav

Inte applicerbar.

B Arkitekturkravsdokument

1 Introduktion

1.1 Syfte

Dokumentets syfte är att beskriva arkitekturkrav för ett webbuppslagsverk.

1.2 Mjukvarans omfattning

Skapa ett sökbart webbuppslagsverk där besökarna kan söka efter, läsa, ändra och lägga till artiklar.

1.3 Definitioner, akronymer och förkortningar

Go - Googles programmeringsspråk som kommer att användas i detta projekt [1].

GoDK - Produkten som utvecklas: **Go D**avid **K**laus Wiki.

1.4 Referenser

[1] Google (2011). The Go Programming Language, *The Go Programming Language*. [www] <http://golang.org/> 2011-02-20.

1.5 Överblick av dokument

2 Systemöversikt

Programmet består av fem olika HTML-sidmallar: Start, Read, Edit, Notfound och Invalid. Sidorna Read, Edit och Notfound är dynamiska och genereras med hjälp av den relevanta textfilen för artikeln.

Användarna tar sig mellan sidorna genom hyperlänkar och omdirigeringar utförda av programmet.

3 Systemkontext

3.1 Webbläsare

Webbläsare som är klient vid anslutning till servern som utgör webbuppslagsverket.

3.2 Filsystem

Filsystemet måste vara läs- och skrivbart för att programmet ska kunna ladda och spara artikelfilerna.

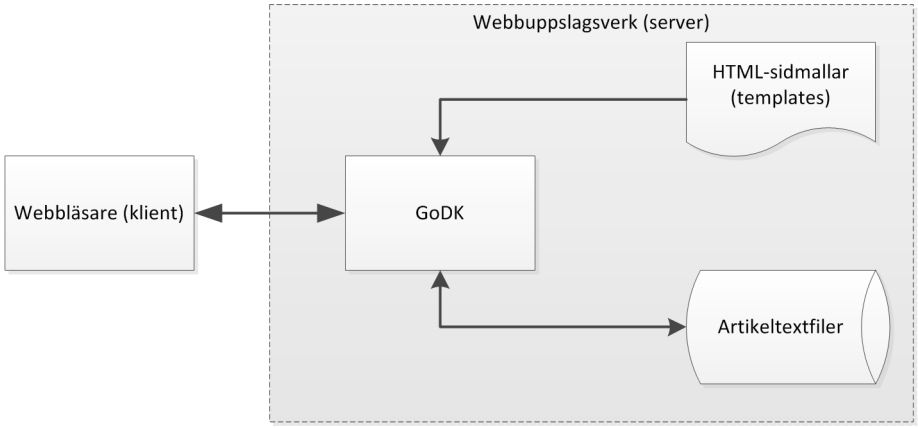
4 Systemets design

4.1 Designmetod

Webbuppslagsverket är serverkomponenten i en klient-server arkitektur.

4.2 Komponentöversikt

En webbläsare ansluter som klient till webbuppslagsverket som är server och består av tre komponenter (*figur B.1*).



Figur B.1 *Övergripande komponentarkitektur.*

Webbuppslagsverket generar, från HTML-sidmallar och artikeltextfilerna, de färdiga sidorna:

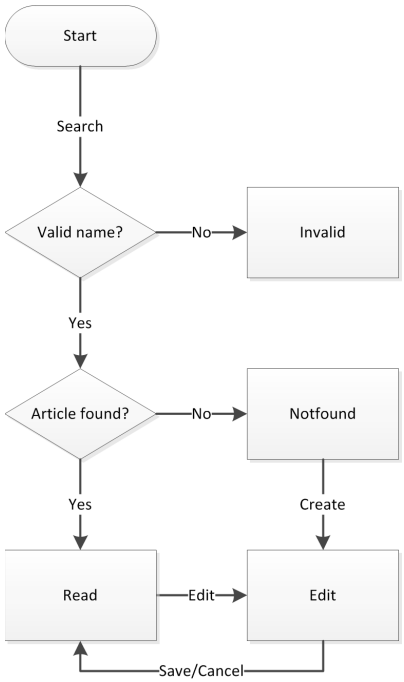
- **Start**: Startsida där man kan söka efter artiklar.
- **Read**: Visar en specifik artikel om den existerar.
- **Edit**: Ändrar en specifik artikel eller skapar ny om den inte existerar.

■ **Notfound**: Denna sida visas om en sökning gjort på en artikeln som inte finns. Möjlighet att skapa artikeln visas.

■ **Invalid**: Denna sida visas om artikelnamnet är otillåtet.

■ **Gemensamma delar**: Delar av webbuppslagsverket som är gemensamma för alla sidor utom Start. Här visas logotyp, länk till start och en sökruta. Detta innebär att det är möjligt att söka från andra sidor.

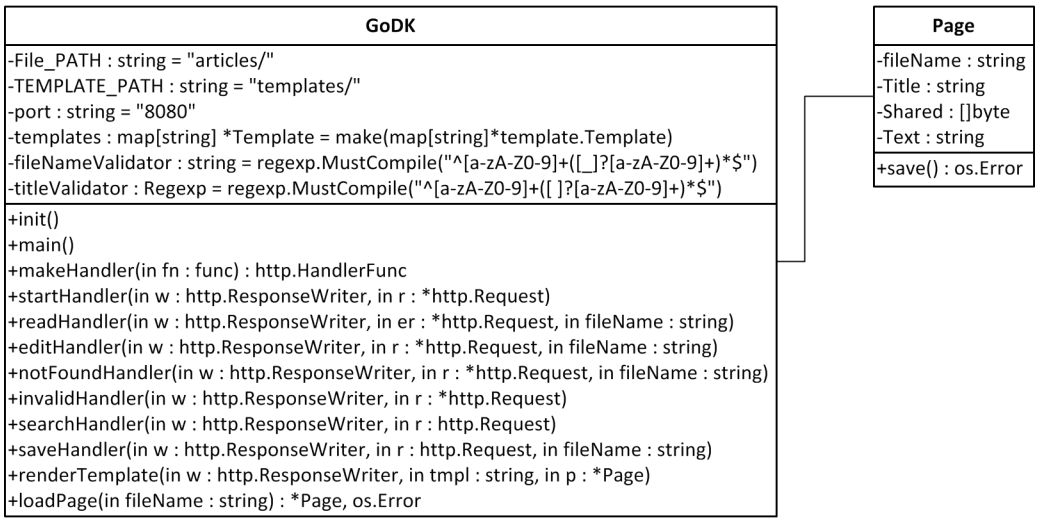
Flödesdiagrammet nedan visar användningen av webbuppslagsverket (*figur B.2)*. Här kan man följa användarscenarion. Det framgår inte i figuren att det finns möjlighet att söka och ta sig till startsidan från alla sidor.



Figur B.2 *Flödesschema för webbuppslagsverket.*

Webbuppslagsverkets komponenter beskrivs nedan.

4.2.1 GoDK
Go programmeringskod som utgör serverdelen av webbuppslagsverket.



Figur B.3 *UML-klassdiagram för GoDK:s paketfil.*

4.2.2 HTML-sidmallar
Det behövs HTML-sidmallar för de färdiga sidorna: Start, Read, Edit, Notfound och Invalid.

4.2.3 Artikeltextfiler
Alla artiklar i webbuppslagsverket sparas i separata textfiler. Dessa utgör en del av innehåller för de dynamiska sidorna Read och Edit.

5 Komponentbeskrivning

5.1 GoDK

5.1.1 Typ
Huvudprogram.

5.1.2 Syfte
Huvudprogrammet som startar webbuppslagsverket.

Funktionskrav i användarkravspecifikation: 3.1.1 – 3.1.5 och 3.1.7 – 3.1.10.

5.1.3 Funktionalitet

Tabell B.1 init

Typ	Funktion
Parametrar	Inga
Returvärde	Inget
Funktionalitet	Utför parsning av HTML-filerna så att de inte behöver läsas från filsystemet mer än en gång per körning av programmet.

Tabell B.2 main

Typ	Funktion
Parametrar	Inga
Returvärde	Inget
Funktionalitet	Kopplar ihop URL:er med Go-funktioner i programmet. Väljer port som ska lyssnas på.

Tabell B.3 makeHandler

Typ	Funktion
Parametrar	`fn func`
Returvärde	`http.HandlerFunc`
Funktionalitet	Returnerar en funktion som hämtar den sökta artikelns filnamn från variabla URL:er och kör funktionen fn om giltigt filnamn annars körs en funktion som omdirigerar till Invalid.

Tabell B.4 startHandler

Typ	Funktion
Parametrar	`w http.ResponseWriter, r *http.Request`
Returvärde	Inget
Funktionalitet	Kör funktionen renderTemplate med w, Start-sidan och nil som argument.

Tabell B.5 readHandler

Typ	Funktion
Parametrar	`w http.ResponseWriter, r *http.Request, fileName string`
Returvärde	Inget
Funktionalitet	Hämtar en Page p med loadPage. Om err returneras omdirigera till Notfound annars kör funktionen renderTemplate med w, Read-sidan och p som argument.

Tabell B.6 editHandler

Typ	Funktion
Parametrar	`w http.ResponseWriter, r *http.Request, fileName string`
Returvärde	Inget
Funktionalitet	Hämtar en Page p med loadPage. Om err returneras skapa en tom Page p och kör funktionen renderTemplate med w, Edit-sidan och p som argument.

Tabell B.7 notFoundHandler

Typ	Funktion
Parametrar	`w http.ResponseWriter, r *http.Request, fileName string`
Returvärde	Inget
Funktionalitet	Skapar en Page p och kör funktionen renderTemplate med w, Notfound-sidan och p som argument.

Tabell B.8 invalidHandler

Typ	Funktion
Parametrar	`w http.ResponseWriter, r *http.Request`
Returvärde	Inget
Funktionalitet	Kör funktionen renderTemplate med w, Invalid-sidan och nil som argument.

Tabell B.9 searchHandler

Typ	Funktion
Parametrar	`w http.ResponseWriter, r *http.Request`
Returvärde	Inget
Funktionalitet	Hämtar den sökta artikelns titel från sökfältet. Kontrollerar om titeln är ogiltig och går till Invalid, annars gör om titeln till filnamn och går till Read.

Tabell B.10 saveHandler

Typ	Funktion
Parametrar	`w http.ResponseWriter, r *http.Request, oldFileName string`
Returvärde	Inget
Funktionalitet	Kontrollerar att artikelns titel är giltig och sparar artikeln. Går till Read om giltig annars till Invalid.

Tabell B.11 renderTemplate

Typ	Funktion
Parametrar	`w http.ResponseWriter, tmpl string, p *Page`
Returvärde	Inget
Funktionalitet	Utför textsubstitution på HTML-sidmallarna (templates) som lägger in det som ska visas.

Tabell B.12 loadPage

Typ	Funktion
Parametrar	`fileName string`
Returvärde	`*Page, os.Error`
Funktionalitet	Hämtar en artikeltext från filsystem till minnet.

Tabell B.13 save

Typ	Metod
Parametrar	Inga
Returvärde	`os.Error`
Funktionalitet	En Page p anropar denna metod som sparar artikeltext till filsystem.

5.1.4 Underkomponenter
Inga.

5.1.5 Beroenden
HTML-sidmallar (templates) måste finnas, artikelfiler måste kunna skapas.

5.1.6 Gränssnitt
Låter användaren läsa, skapa och ändra artiklar som ligger i filsystemet.

5.1.7 Systemkrav
Linux.

5.1.8 Referenser
Inga.

5.1.9 Bearbetning
Se *figur B.1*.

5.1.10 Data

Namn	Typ	Beskrivning
`FILE_PATH`	`string`	Sökvägen till artikeltexterna.
`TEMPLATE_PATH`	`string`	Sökvägen till HTML-sidmallarna.
`port`	`string`	En strängrepresentation av portnumret som programmet kommer att lyssna på. (Standard är "8080").
`templates`	`map[string] *Template`	En datastruktur för HTML-sidmallar.
`fileNameValidator`	`Regexp`	Ett reguljärt uttryck för tillåtna filnamn.
`titleValidator`	`Regexp`	Ett reguljärt uttryck för tillåtna artikeltitlar.

5.2 Start

5.2.1 Typ
HTML-grundsida (template).

5.2.2 Syfte
HTML-grundsida som GoDK använder som startsida (Home) varifrån användaren kan söka efter artiklar.

Funktionskrav i användarkravspecifikation: 3.1.1.

5.2.3 Funktionalitet
Inga funktioner.

5.2.4 Underkomponenter
Inga.

5.2.5 Beroenden
GoDK.

5.2.6 Gränssnitt
Söker artiklar via GoDK.

5.2.7 Systemkrav
Webbläsare.

5.2.8 Referenser
Inga.

5.2.9 Bearbetning
Se *figur B.1*.

5.2.10 Data
Ingen.

5.3 Read

5.3.1 Typ
HTML-grundsida (template).

5.3.2 Syfte
HTML-grundsida som GoDK använder för att visa artiklar för användaren.

Funktionskrav i användarkravspecifikation: 3.1.1 och 3.1.7

5.3.3 Funktionalitet
Inga funktioner.

5.3.4 Underkomponenter
Inga.

5.3.5 Beroenden
GoDK.

5.3.6 Gränssnitt
Har en länk som kan ta användaren till Edit-sidan med hjälp av GoDK.

5.3.7 Systemkrav
Webbläsare.

5.3.8 Referenser
Inga.

5.3.9 Bearbetning
Se *figur B.1*.

5.3.10 Data
Ingen.

5.4 Edit

5.4.1 Typ
HTML-grundsida (template).

5.4.2 Syfte
HTML-grundsida som GoDK använder för att låta användaren ändra artiklar.

Funktionskrav i användarkravspecifikation: 3.1.1 – 3.1.5 och 3.1.7.

5.4.3 Funktionalitet
Inga funktioner.

5.4.4 Underkomponenter
Inga.

5.4.5 Beroenden
GoDK.

5.4.6 Gränssnitt
Har en länk som låter användaren spara artikeln med hjälp av GoDK.

5.4.7 Systemkrav
Webbläsare.

5.4.8 Referenser
Inga.

5.4.9 Bearbetning
Se *figur B.1*.

5.4.10 Data
Ingen.

5.5 Notfound

5.5.1 Typ
HTML-grundsida (template).

5.5.2 Syfte
HTML-grundsida som GoDK använder för att visa att den sökta artikeln ej hittades.

Funktionskrav i användarkravspecifikation: 3.1.1 och 3.1.6 – 3.1.7.

5.5.3 Funktionalitet
Inga funktioner.

5.5.4 Underkomponenter
Inga.

5.5.5 Beroenden
GoDK.

5.5.6 Gränssnitt
Har en länk som tillåter användaren att komma till Edit-sidan, med hjälp av GoDK, för att skapa en ny artikel.

5.5.7 Systemkrav
Webbläsare.

5.5.8 Referenser
Inga.

5.5.9 Bearbetning
Se *figur B.1*.

5.5.10 Data
Ingen.

5.6 Invalid

5.6.1 Typ
HTML-grundsida (template).

5.6.2 Syfte
HTML-grundsida som GoDK använder för att visa att sökningen var ottilåten.

Funktionskrav i användarkravspecifikation: 3.1.1 och 3.1.7.

5.6.3 Funktionalitet
Inga funktioner.

5.6.4 Underkomponenter
Inga.

5.6.5 Beroenden
GoDK.

5.6.6 Gränssnitt
Inget.

5.6.7 Systemkrav
Webbläsare.

5.6.8 Referenser

Inga.

5.6.9 Bearbetning
Se *figur B.1*.

5.6.10 Data
Ingen.

5.7 Artikleltextfiler

5.6.1 Typ
Textfil.

5.6.2 Syfte
Lagrar artikeltext och -format.

Funktionskrav i användarkravspecifikation: 3.1.2 – 3.1.5 och 3.1.9.

5.6.3 Funktionalitet
Inga funktioner.

5.6.4 Underkomponenter
Inga.

5.6.5 Beroenden
GoDK.

5.6.6 Gränssnitt
Inget.

5.6.7 Systemkrav
Filsystem.

5.6.8 Referenser
Inga.

5.6.9 Bearbetning
Se *figur B.1*.

5.6.10 Data
Ingen.

6 Genomförbarhet och tidsuppskattningar

Tabell B.14 Komponenternas versionstillhörighet

Komponent	Standard	Avancerad
GoDK	✓	
Gemensamma delar	✓	
Start	✓	
Read	✓	
Edit	✓	
Notfound	✓	
Invalid	✓	

Tabell B.15 Tidsuppskattningar för komponenternas utveckling

Komponent	Beskrivning	Planerad tid (h)
GoDK	Huvudprogrammet som startar webbuppslagsverket.	25
Gemensamma delar	Delar av webbsidan som är gemensamma för alla sidor utom Start.	10
Start	Startsida där man kan söka efter artiklar.	2
Read	HTML-grundsida som GoDK använder för att visa artiklar för användaren.	5
Edit	Ändrar en specifik artikel eller skapar ny om den inte existerar.	10
Notfound	Denna sida visas om en sökning gjort på en artikeln som inte finns. Möjlighet att skapa artikeln visas.	2
Invalid	Denna sida visas om artikelnamnet är otillåtet.	2

Tabell B.15 Spårbarhetsmatris mellan Användarkrav (URD) och Arkitekturkrav (ADD)

		URD									
		3.1.1	3.1.2	3.1.3	3.1.4	3.1.5	3.1.6	3.1.7	3.1.8	3.1.9	3.1.10
ADD	5.1	✓	✓	✓	✓	✓		✓	✓	✓	✓
	5.2	✓									
	5.3	✓						✓			
	5.4	✓	✓	✓	✓	✓		✓			
	5.5	✓					✓	✓			
	5.6	✓						✓			
	5.7		✓	✓	✓	✓				✓	

C GoDK programkod

GoDK.go

```
1  // Copyright 2011 David Falk and Klaus Nicosia. All rights reserved.
2
3  // The main package of GoDK.
4  package main
5
6  import (
7     "http"
8     "io/ioutil"
9     "os"
10    "template"
11    "regexp"
12    "strings"
13    "fmt"
14 )
15
16 // File path where the articles are located.
17 const FILE_PATH = "articles/"
18
19 // File path where the templates are located.
20 const TEMPLATE_PATH = "templates/"
21
22 // Standard port to be used by GoDK.
23 var port = "8080"
24
25 // Map for storing *Template for the html files that are used by GoDK.
26 var templates = make(map[string]*template.Template)
27
28 // A regex to check if a filename is valid.
29 var fileNameValidator = regexp.MustCompile("^[a-zA-Z0-9]+([_]?[a-zA-Z0-9]+)*$")
30
31 // A regex to check if a title is valid.
32 var titleValidator = regexp.MustCompile("^[a-zA-Z0-9]+([ ]?[a-zA-Z0-9]+)*$")
33
34 // Initialization of the html files to be done before main method.
35 func init() {
36    initHtmlFiles()
37 }
38  // Function initHtmlFiles (re-)initializes the HTML files.
39 func initHtmlFiles() {
40    for _, tmpl := range []string{"start", "read", "edit", "notfound",
41          "invalid","redirect"} {
42       templates[tmpl] = template.MustParseFile(TEMPLATE_PATH+tmpl+".html", nil)
43    }
44 }
45
46 // Function main listens to the specific port and starts a new goroutine with
47 // the menu.
48 func main() {
49    http.HandleFunc("/", startHandler)
50    http.HandleFunc("/read/", makeHandler(readHandler))
51    http.HandleFunc("/edit/", makeHandler(editHandler))
52    http.HandleFunc("/notfound/", makeHandler(notFoundHandler))
53    http.HandleFunc("/invalid/", invalidHandler)
54    http.HandleFunc("/redirect/", redirectHandler)
55
56    http.Handle("/go/", http.FileServer("/home/david/gowiki/templates/", "/go/"))
57
58    http.HandleFunc("/search/", searchHandler)
59    http.HandleFunc("/save/", makeHandler(saveHandler))
60    go printInfo()
61    http.ListenAndServe(":" + port, nil)
62 }
63
64 // Prints welcome message and lets the user choose from a menu.
65 func printInfo() {
```

```
66     var i int
67     fmt.Println("Welcome to Go David Klaus (Awesome Wiki) version 1.0")
68     fmt.Println("Listening to port: " + port)
69
70     for {
71        writeMenu()
72           _,err := fmt.Scanf("%d", &i)
73        if err == nil {
74           doSelection(i)
75        }
76     }
77  }
78
79  // Writes the menu of options.
80  func writeMenu() {
81     fmt.Println("   ***** Menu *****")
82     fmt.Println("0: Exit the program")
83     fmt.Println("1: Change Port - (Not Implemented)")
84     fmt.Println("2: Reinitialize HTML files")
85     fmt.Println("3: Delete article")
86  }
87
88  // Executes the selection that the user has chosen.
89  func doSelection(i int) {
90     switch i {
91     case 0:
92        os.Exit(0)
93     case 1:
94        fmt.Println("This will not work")
95     case 2:
96        initHtmlFiles()
97        fmt.Println("HTML files reinitialized")
98     case 3:
99        chooseArticleDelete()
100    default:
101
102    }
103 }
104
105 // The function makeHandler takes a function as an argument and returns a
106 // function that will be used by http.HandleFunc when a third parameter
107 // (filename) is necessary.
108 func makeHandler(fn func (http.ResponseWriter,
109       *http.Request, string)) http.HandlerFunc {
110    return func(w http.ResponseWriter, r *http.Request) {
111       path := strings.Split(r.URL.Path,"/",3)
112       fileName := path[2]
113       if !checkFileName(fileName) {
114          http.Redirect(w, r, "/invalid", http.StatusFound)
115          return
116       }
117       fn(w, r, fileName)
118    }
119 }
120
121 // Handler that shows the startpage.
122 func startHandler(w http.ResponseWriter, r *http.Request) {
123    renderTemplate(w, "start", nil)
124 }
125
126 // Handler that gets the article to be shown and generates the page to be
127 // viewed.
128 func readHandler(w http.ResponseWriter, r *http.Request, fileName string) {
129    p, err := loadPage(fileName)
130    if err != nil {
131       http.Redirect(w, r, "/notfound/"+fileName, http.StatusFound)
132       return
133    }
134    regex := regexp.MustCompile("{REDIRECT}.*{/REDIRECT}")
135    if(regex.MatchString(p.Text)) {
136       fmt.Println(p.Text)
137       match := regex.FindString(p.Text)[len("{REDIRECT}"):]
138       fmt.Println(match)
```

```
139         match = strings.TrimRight(match,"{/REDIRECT}")
140         fmt.Println(match)
141         http.Redirect(w, r, "/redirect/"+fileName+"/"+titleToFileName(match),
142               http.StatusFound)
143         return
144      }
145      p.Text = changeText(p.Text)
146      renderTemplate(w, "read", p)
147   }
148
149   // Handler that gets the article to be edited and generates the page to be
150   // viewed.
151   func editHandler(w http.ResponseWriter, r *http.Request, fileName string) {
152      p, err := loadPage(fileName)
153      title := fileNameToTitle(fileName)
154      if err != nil {
155         p = &Page{Title: title, FileName: fileName}
156      }
157      renderTemplate(w, "edit", p)
158   }
159
160   // Handler that gets the article to be shown and generates the page to be
161   // viewed.
162   func notFoundHandler(w http.ResponseWriter, r *http.Request, fileName string) {
163      title := fileNameToTitle(fileName)
164      p := &Page{FileName: fileName, Title: title}
165      renderTemplate(w, "notfound", p)
166   }
167
168   // Handler that shows the invalid-page.
169   func invalidHandler(w http.ResponseWriter, r *http.Request) {
170      renderTemplate(w, "invalid", nil)
171   }
172
173   // Handler that redirects the user to the page that the previous article
174   // redirects to.
175   func redirectHandler(w http.ResponseWriter, r *http.Request){
176      fileNames := strings.Split(r.URL.Path[len("/redirect/"):], "/", -1)
177      renderTemplate(w, "redirect", &Page{Title: fileNameToTitle(fileNames[1]),
178            FileName: fileNames[1], Text: fileNames[0]})
179   }
180
181   // Handler that searches after an article and if the title is valid redirects
182   // to read else it redirects to invalid.
183   func searchHandler(w http.ResponseWriter, r *http.Request) {
184      temptitle := r.FormValue("title")
185      title := strings.TrimSpace(temptitle) //removes leading and trailing spaces
186      if (checkTitle(title)) {
187         fileName := titleToFileName(title)
188         http.Redirect(w, r, "/read/" + fileName, http.StatusFound)
189      } else {
190         http.Redirect(w, r, "/invalid", http.StatusFound)
191      }
192   }
193
194   // Handler that saves the article that has been edited, then redirects to that
195   // article.
196   func saveHandler(w http.ResponseWriter, r *http.Request, oldFileName string) {
197      title := strings.TrimSpace(r.FormValue("title"))
198      text := r.FormValue("text")
199      if checkTitle(title) {
200         title = firstLetterToUpper(title)
201         fileName := titleToFileName(title)
202         p := &Page{Title: title, FileName: fileName, Text: text}
203         err := p.save()
204         if err != nil {
205            http.Error(w, err.String(), http.StatusInternalServerError)
206            return
207         }
208         http.Redirect(w, r, "/read/"+fileName, http.StatusFound)
209      } else {
210         http.Redirect(w, r, "/invalid", http.StatusFound)
211      }
```

```
212 }
213
214 // Takes a *Page and the name of the html file and uses template.Execute to
215 // show the data from the Page in the generated html-file and renders
216 // the html file.
217 func renderTemplate(w http.ResponseWriter, tmpl string, p *Page) {
218    err := templates[tmpl].Execute(w, p)
219    if err != nil {
220       http.Error(w, err.String(), http.StatusInternalServerError)
221    }
222 }
223
224 // Gets a Page from an article file, returns any errors that may exist in
225 // the second return parameter.
226 func loadPage(fileName string) (*Page, os.Error) {
227    tempFileName, err:= findName(fileName)
228    if err != nil {
229       return nil, err
230    }
231    fullFileName := FILE_PATH+tempFileName+".txt"
232    text, err := ioutil.ReadFile(fullFileName)
233    if err != nil {
234       return nil, err
235    }
236    shared, err := ioutil.ReadFile(TEMPLATE_PATH+"shared.html")
237    if err != nil {
238       return nil, err
239    }
240    title := fileNameToTitle(tempFileName)
241    return &Page{FileName: tempFileName, Title: title, Text: string(text),
242          Shared: shared}, nil
243 }
244
245 // Checks if the filename is correct.
246 func checkFileName(fileName string) bool {
247    return fileNameValidator.MatchString(fileName)
248 }
249
250 // Checks if the title is correct.
251 func checkTitle(title string) bool {
252    return titleValidator.MatchString(title)
253 }
254
255 // Takes a title as parameter and returns the filename for the article.
256 func titleToFileName(title string) string {
257    return strings.Replace(title, " ", "_", -1)
258 }
259
260 // Takes a filename as parameter and returns the title for the article.
261 func fileNameToTitle(fileName string) string {
262    return strings.Replace(fileName, "_", " ", -1)
263 }
264
265 // Sets the first letter of a string to uppercase and returns the new string.
266 func firstLetterToUpper(word string) string {
267    if (word[0] >= 97) {
268       firstLower := string(word[0])
269       firstUpper := string(word[0] - 32)
270       word = strings.Replace(word, firstLower, firstUpper, 1)
271    }
272    return word
273 }
274
275 // A case-insensitive check if an article exists with the filename. If the
276 // article exists the name of that article is returned.
277 func findName(fileName string) (string, os.Error) {
278    cmpfile := strings.ToLower(fileName)+".txt"
279    fileinfoarray, err := ioutil.ReadDir(FILE_PATH);
280    if err != nil {
281       return "", err
282    }
283    for i:=0 ; i<len(fileinfoarray) ; i++ {
284       if strings.ToLower(fileinfoarray[i].Name)==cmpfile {
```

```
285            return strings.TrimRight(fileinfoarray[i].Name,".txt"), nil
286         }
287      }
288      return "", os.NewError("File not found")
289   }
290
291   // Removes all files with this filename.
292   func removeOld(fileName string) os.Error {
293      cmpfile := strings.ToLower(fileName)+".txt"
294      fileinfoarray, err := ioutil.ReadDir(FILE_PATH);
295      if err != nil {
296         return err
297      }
298      for i:=0 ; i<len(fileinfoarray) ; i++ {
299         if strings.ToLower(fileinfoarray[i].Name)==cmpfile {
300            if fileinfoarray[i].Name!=(fileName+".txt") {
301               err = os.Remove(FILE_PATH+fileinfoarray[i].Name)
302               if err != nil {
303                  return err
304               }
305            }
306         }
307      }
308      return nil
309   }
310
311   // Menu for the user to choose which article to delete.
312   func chooseArticleDelete() {
313      var i int
314      fileinfoarray, err := ioutil.ReadDir(FILE_PATH);
315      fmt.Println()
316      if err!=nil {
317         fmt.Println("Cannot Get Articles")
318      } else {
319         fmt.Println(i,": Return to Menu")
320         fmt.Println("Articles(By Name):")
321         for i:=0 ; i<len(fileinfoarray) ; i++ {
322            fmt.Println(i+1, ": " +
323                  fileNameToTitle(strings.TrimRight(fileinfoarray[i].Name,
324                  ".txt")))
325         }
326         fmt.Println()
327         fmt.Println("Enter number of the Article to delete")
328         fmt.Scanf("%d", &i)
329         if(i>0 && i<len(fileinfoarray)) {
330            err = deleteArticle(fileinfoarray[i-1].Name)
331            if err != nil {
332               fmt.Println("Could not delete article " +
333                     fileNameToTitle(strings.TrimRight(
334                     fileinfoarray[i-1].Name,".txt")))
335               return
336            }
337            fmt.Println("Article " + fileNameToTitle(strings.TrimRight(
338                  fileinfoarray[i-1].Name,".txt"))+" deleted")
339            fmt.Println()
340         } else if i!=0 {
341            fmt.Println("No article with that number")
342         }
343      }
344   }
345
346   // Delete article.
347   func deleteArticle(article string) os.Error {
348      return os.Remove(FILE_PATH+article)
349   }
350
351   // Converts text so it can be viewed correctly as html when showing an article.
352   func changeText(text string) string {
353      text = strings.Replace(text,"[b]","<b>",-1)
354      text = strings.Replace(text,"[/b]","</b>",-1)
355      text = strings.Replace(text,"[quote]","<blockquote>",-1)
356      text = strings.Replace(text,"[/quote]","</blockquote>",-1)
357      text = strings.Replace(text,"[code]","<code>",-1)
```

```
358     text = strings.Replace(text,"[/code]","</code>",-1)
359     text = strings.Replace(text,"[u]","<u>",-1)
360     text = strings.Replace(text,"[/u]","</u>",-1)
361     text = strings.Replace(text,"[i]","<i>",-1)
362     text = strings.Replace(text,"[/i]","</i>",-1)
363     text = strings.Replace(text,"[h2]","<h2>",-1)
364     text = strings.Replace(text,"[/h2]","</h2>",-1)
365     text = strings.Replace(text,"[h3]","<h3>",-1)
366     text = strings.Replace(text,"[/h3]","</h3>",-1)
367     text = strings.Replace(text,"\n","<br>",-1)
368     text = strings.Replace(text,"[img]","<img src=\"",-1)
369     text = strings.Replace(text,"[/img]","\" alt=\"\" />",-1)
370 
371     text = strings.Replace(text,"[url]","<a href=\"",-1)
372     text = strings.Replace(text,"[urlsep]","\">",-1)
373     text = strings.Replace(text,"[/url]","</a>",-1)
374     text = wikiLinks(text)
375     return text
376 }
377 
378 // Gets a string and checks if links exists in the string. Replaces
379 // these with html href.
380 func wikiLinks(s string) string{
381     regex := regexp.MustCompile("{{[^{}]*}}")
382     for regex.MatchString(s) {
383         new_s := regex.FindString(s)
384         new2_s := strings.TrimRight(new_s,"}}")[len("{{"):]
385         s = strings.Replace(s, new_s, "<a href=\"/read/" +
386               titleToFileName(new2_s) + "\">" + new2_s + "</a>", 1)
387     }
388     return s
389 }
```

Page.go

```
1  // Copyright 2011 David Falk and Klaus Nicosia. All rights reserved.
2  
3  // The main package of GoDK.
4  package main
5  
6  import (
7     "html"
8     "io/ioutil"
9     "os"
10 )
11 
12 // Page is a struct that is used to send data with the templates.execute method.
13 type Page struct {
14     FileName string
15     Title    string
16     Shared   []byte
17     Text     string
18 }
19 
20 // The method save saves the struct to file and returns os.Error
21 // if something went wrong.
22 func (p *Page) save() os.Error {
23     filename := FILE_PATH+p.FileName+".txt"
24     text := html.EscapeString(p.Text)
25     err := ioutil.WriteFile(filename, []byte(text), 0600)
26     if err != nil {
27         return err
28     }
29     err = removeOld(p.FileName)
30     if err != nil {
31         return err
32     }
33     return nil
34 }
```

start.html

```
<html>
<head>
    <link rel="stylesheet" type="text/css" href="/go/godk.css" />
    <title>GoDK</title>
</head>
<body>
    <div class="start_div">
        <a href="/"><img src="/go/godk_logotyp.png" class="logotype" width="250"
            height="100"/></a>
        <form action="/search/" method="POST">
            <div>
                <input type="text" name="title">
                <input type="submit" value="Search">
            </div>
        </form>
    </div>
</body>
</html>
```

shared.html

```
<div class="top_div">
    <form action="/search/" method="POST">
    <div class="logotype_div">
        <a href="/"><img src="/go/godk_logotyp.png" class="logotype"
        width="250" height="100"/></a>
    </div>
    <div>
        <input type="text" name="title">
        <input type="submit" value="Search">
    </div>
    </form>
</div>
<div class="clear_div"></div>
<div class="left_div">
    <p><a href="/edit/New_Article">New Article</a></p>
</div>
```

read.html

```
<html>
<head>
    <title>{Title}</title>
    <link rel="stylesheet" type="text/css" href="/go/godk.css" />
</head>
<body>
    {Shared}
    <div class="content_div">
    <form action="/edit/{FileName}" method="POST">
    <div><input type="submit" value="Edit Article"></div>
    </form>
    <h1>{Title}</h1>
    <div>{Text}</div>
    </div>
</body>
</html>
```

edit.html

```
<html>
<head>
    <title>Edit Article {Title}</title>
    <link rel="stylesheet" type="text/css" href="/go/godk.css" />
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="/go/editor/ed.js"></script>
</head>
<body>
    {Shared}
```

```
    <div class="content_div">
    <form action="/read/{FileName}" method="POST">
        <div><input type="submit" value="Read Article"></div>
    </form>
    <form action="/save/{FileName}" method="POST">
        <div>
        <input type="text" name="title" value="{Title}" />
        <input type="submit" value="Save Article">
        </div>
        <p>
        <script>edToolbar('text'); </script>
        <textarea name="text" rows="80" cols="250" id="text"
            class="ed">{Text}</textarea>
        </p>
    </form>
    </div>
<body/>
</html>
```

notfound.html

```
<head>
    <link rel="stylesheet" type="text/css" href="/go/godk.css" />
    <title>Article not found</title>
</head>
<body>
    <div class="top_div">
        <form action="/search/" method="POST">
            <div class="logotype_div">
                <a href="/"><img src="/go/godk_logotyp.png" class="logotype"
                    width="250" height="100"/></a>
            </div>
            <div>
                <input type="text" name="title">
                <input type="submit" value="Search">
            </div>
        </form>
    </div>
    <div class="clear_div"></div>
    <div class="left_div">
        <p><a href="/edit/New_Article">New Article</a></p>
    </div>
    <div class="content_div">
        <h1>{Title} was not found</h1>
        <a href="/edit/{FileName}">Create article {Title}</a>
    </div>
</body>
```

invalid.html

```
<html>
<head>
    <link rel="stylesheet" type="text/css" href="/go/godk.css" />
    <title>Invalid Article Name</title>
    </head>
<body>
    <div class="top_div">
    <form action="/search/" method="POST">
        <div class="logotype_div">
            <a href="/"><img src="/go/godk_logotyp.png" class="logotype"
                width="250" height="100"/></a></div>
            <div>
                <input type="text" name="title">
                <input type="submit" value="Search">
        </div>
    </form>
    </div>
    <div class="clear_div"></div>
    <div class="left_div">
        <p><a href="/edit/New_Article">New Article</a></p>
    </div>
    <div class="content_div">
```

```
    <h1>Invalid Article Name</h1>
    Article Name can only contain a-zA-Z0-9 and space (cannot start or end
     with space and two or more spaces in a row is not allowed).
    </div>
</body>
</html>
```

redirect.html

```
<head>
    <link rel="stylesheet" type="text/css" href="/go/godk.css" />
    <meta http-equiv="refresh" content="3;url=/read/{FileName}">
    <title>Redirecting</title>
</head>
<body>
    <div class="top_div">
        <form action="/search/" method="POST">
            <div class="logotype_div">
                <a href="/"><img src="/go/godk_logotyp.png" class="logotype"
                    width="250" height="100"/></a>
            </div>
            <div>
                <input type="text" name="title">
                <input type="submit" value="Search">
            </div>
        </form>
    </div>
    <div class="clear_div"></div>
    <div class="left_div">
        <p><a href="/edit/New_Article">New Article</a></p>
    </div>
    <div class="content_div">
    <form action="/edit/{Text}" method="POST">
    <div><input type="submit" value="Edit Article"></div>
    </form>
    <h1>Redirect</h1><br>
    Redirecting to {Title}
    </div>
</body>
```

godk.css

```
.start_div {
    margin-top:100px;
    text-align:center;
}

.top_div {
    margin-left:none;
    margin-right:none;
    width:100%;
    background-color:#d0f0f6;
    text-align:left;
}

.clear_div {
    clear:both;
}

.logotype_div {
    position:fixed;
    margin:none;
    left:0px;
    top:0px;
    width:260px;
    background-color:#FFFFFF;
    text-align:left;
}

.top_div {
    margin-left:none;
    margin-right:none;
```

```
    background-color:#FFFFFF;
    text-align:right;
}

.left_div {
    position:fixed;
    top:100px;
    width:218px;
    height:100%;
    margin:none;
    background-color:#FFFFFF;
    text-align:left;
    padding:16px;
}

.content_div {
    padding:0px 16px 16px 16px;
    margin-left:260px;
    height:100%;
    border-left:1px solid gray;
    border-top:1px solid gray;
}

h1 {
    color:#000000;
    font:48px Georgia, "Times New Roman", Times, serif;
    margin:0px;
    text-align:left;
}

h2 {
    color:#000000;
    font:28px Georgia, "Times New Roman", Times, serif;
    margin:5px 0px 2px;
    text-align:left;
}

h3 {
    color:#000000;
    font:23px Georgia, "Times New Roman", Times, serif;
    margin:5px 0px 2px;
    text-align:left;
}

body {
    color:#000000;
    font:17px Georgia, "Times New Roman", Times, serif;
    text-align:left;
}

a:link {
    color:#0000FF;
    font:17px Georgia, "Times New Roman", Times, serif;
    text-align:left;
}

a:visited {
    color:#9900FF;
    font:17px Georgia, "Times New Roman", Times, serif;
    text-align:left;
}

a:hover {
    color:#FF00CC;
    font:17px Georgia, "Times New Roman", Times, serif;
    text-align:left;
}

a:active {
    color:#33FF00;
    font:17px Georgia, "Times New Roman", Times, serif;
    text-align:left;
}
```

```
104
105 img {
106     margin:12px;
107     border-style: none;
108     font:12px Georgia, "Times New Roman", Times, serif;
109 }
110
111 img.logotype {
112     border-style: none;
113     font:12px Georgia, "Times New Roman", Times, serif;
114     margin:1px 1px 1px 1px;
115 }
```

ed.js

```
  1 /*
  2  * Name: Javascript Textarea HTML Editor
  3  * Author: Balakrishnan
  4  * License: Free
  5  * URL: http://www.corpocrat.com
  6  *
  7  *
  8  * Edited by David Falk and Klaus Nicosia
  9  * Version: 1.4
 10  * Last Modified Date: 2011-04-25
 11  */
 12
 13 var textarea;
 14 var content;
 15 document.write("<link href=\"/go/editor/styles.css\" rel=\"stylesheet\" "
 16     + "type=\"text/css\">");
 17
 18
 19 function edToolbar(obj) {
 20     document.write("<img class=\"button\" "
 21         + "src=\"/go/editor/images/bold.gif\" name=\"btnBold\" "
 22         + "title=\"Bold\" onClick=\"doAddTags('[b]','[/b]','" + obj + "')\">");
 23
 24     document.write("<img class=\"button\" "
 25         + "src=\"/go/editor/images/italic.gif\" name=\"btnItalic\" "
 26         + "title=\"Italic\" onClick=\"doAddTags('[i]','[/i]','" + obj
 27         + "')\">");
 28
 29     document.write("<img class=\"button\" "
 30         + "src=\"/go/editor/images/underline.gif\" name=\"btnUnderline\" "
 31         + "title=\"Underline\" onClick=\"doAddTags('[u]','[/u]','" + obj
 32         + "')\">");
 33
 34     document.write("<img class=\"button\" "
 35         + "src=\"/go/editor/images/redirect2.gif\" name=\"btnRedirect\" "
 36         + "title=\"Redirect\" onClick=\"doAddTags('{REDIRECT}','{/REDIRECT} "
 37         + "','" + obj + "')\">");
 38
 39     document.write("<img class=\"button\" "
 40         + "src=\"/go/editor/images/link.gif\" name=\"btnLink\" "
 41         + "title=\"Insert Link\" onClick=\"doURL('" + obj + "')\">");
 42
 43     document.write("<img class=\"button\" "
 44         + "src=\"/go/editor/images/link.gif\" name=\"btnArticleLink\" "
 45         + "title=\"Insert Article Link\" onClick=\"doAddTags('{{','}}','"
 46         + obj + "')\">");
 47
 48     document.write("<img class=\"button\" "
 49         + "src=\"/go/editor/images/image.gif\" name=\"btnPicture\" "
 50         + "title=\"Insert Picture\" onClick=\"doImage('" + obj + "')\">");
 51
 52     document.write("<img class=\"button\" "
 53         + "src=\"/go/editor/images/header2.gif\" name=\"btnHeader_2\" "
 54         + "title=\"Header 2\" onClick=\"doAddTags('[h2]','[/h2]','" + obj
 55         + "')\">");
 56
 57     document.write("<img class=\"button\" "
 58         + "src=\"/go/editor/images/header3.gif\" name=\"btnHeader_3\" "
```

```
        + "title=\"Header 3\" onClick=\"doAddTags('[h3]','[/h3]','" + obj
        + "')\">");

    document.write("<img class=\"button\" "
        + "src=\"/go/editor/images/quote.gif\" name=\"btnQuote\" "
        + "title=\"Quote\" onClick=\"doAddTags('[quote]','[/quote]','" + obj
        + "')\">");

    document.write("<img class=\"button\" "
        + "src=\"/go/editor/images/code.gif\" name=\"btnCode\" "
        + "title=\"Code\" onClick=\"doAddTags('[code]','[/code]','" + obj
        + "')\">");

    document.write("<br>");
}

function doImage(obj) {
    textarea = document.getElementById(obj);
    var url = prompt('Enter the Image URL:','http://');

    var img1 = '[img]';
    var img2 = '[/img]';

    var scrollTop = textarea.scrollTop;
    var scrollLeft = textarea.scrollLeft;

    if (url != '' && url != null) {
        if (document.selection) {
            textarea.focus();
            var sel = document.selection.createRange();
            sel.text = img1 + url + img2;
        } else {
            var len = textarea.value.length;
            var start = textarea.selectionStart;
            var end = textarea.selectionEnd;
            var sel = textarea.value.substring(start, end);
            var rep = img1 + url + img2;
            textarea.value =  textarea.value.substring(0,start) + rep
                + textarea.value.substring(end,len);
            textarea.scrollTop = scrollTop;
            textarea.scrollLeft = scrollLeft;
        }
    }
}

function doURL(obj) {
    var sel;
    textarea = document.getElementById(obj);
    var url = prompt('Enter the URL:','http://');
    var scrollTop = textarea.scrollTop;
    var scrollLeft = textarea.scrollLeft;

    var url1 = '[url]';
    var url2 = '[urlsep]';
    var url3 = '[/url]';

    if (url != '' && url != null) {
        if (document.selection) {
            textarea.focus();
            var sel = document.selection.createRange();

            if (sel.text == "") {
                sel.text = url1 + url + url2 + url + url3;
            } else {
                sel.text = url1 + url + url2 + sel.text + url3;
            }
        } else {
            var len = textarea.value.length;
            var start = textarea.selectionStart;
            var end = textarea.selectionEnd;
            var sel = textarea.value.substring(start, end);

        if (sel == "") {
```

```
            sel=url;
        } else {
            var sel = textarea.value.substring(start, end);
            }
            var rep = url1 + url + url2 + sel + url3;
            textarea.value = textarea.value.substring(0,start) + rep
                + textarea.value.substring(end,len);
            textarea.scrollTop = scrollTop;
            textarea.scrollLeft = scrollLeft;
        }
    }
}

function doAddTags(tag1,tag2,obj) {
    textarea = document.getElementById(obj);
    if (document.selection){
        // Code for IE.
        textarea.focus();
        var sel = document.selection.createRange();
        sel.text = tag1 + sel.text + tag2;
    } else {
        var len = textarea.value.length;
        var start = textarea.selectionStart;
        var end = textarea.selectionEnd;
        var scrollTop = textarea.scrollTop;
        var scrollLeft = textarea.scrollLeft;
        var sel = textarea.value.substring(start, end);
        var rep = tag1 + sel + tag2;
        textarea.value =  textarea.value.substring(0,start) + rep
            + textarea.value.substring(end,len);
        textarea.scrollTop = scrollTop;
        textarea.scrollLeft = scrollLeft;
    }
}

function doList(tag1,tag2,obj) {
    textarea = document.getElementById(obj);
    if (document.selection) {
        // Code for IE.
        textarea.focus();
        var sel = document.selection.createRange();
        var list = sel.text.split('\n');

        for(i = 0; i < list.length; i++) {
            list[i] = '<li>' + list[i] + '</li>';
        }
        sel.text = tag1 + '\n' + list.join("\n") + '\n' + tag2;

        } else {
        var len = textarea.value.length;
        var start = textarea.selectionStart;
        var end = textarea.selectionEnd;
        var i;
        var scrollTop = textarea.scrollTop;
        var scrollLeft = textarea.scrollLeft;
        var sel = textarea.value.substring(start, end);
        var list = sel.split('\n');
        for(i = 0; i < list.length; i++) {
            list[i] = '<li>' + list[i] + '</li>';
        }
        var rep = tag1 + '\n' + list.join("\n") + '\n' +tag2;
        textarea.value =  textarea.value.substring(0,start) + rep
            + textarea.value.substring(end,len);
        textarea.scrollTop = scrollTop;
        textarea.scrollLeft = scrollLeft;
    }
}
```

style.css

```
.editor {
    font-family: verdana, arial, sans-serif;
    font-size:13px;
    padding: 5px;
    background: #d5e8f9;
    border: 1px solid #e5ecf9;
}

.button {
    margin: 1px;
    padding: 2px;
}

.button:hover {
    opacity:.60;
    filter: alpha(opacity=60);
    -moz-opacity: 0.60;
}

.ed {
    width: 400px;
    height: 150px;
}
```