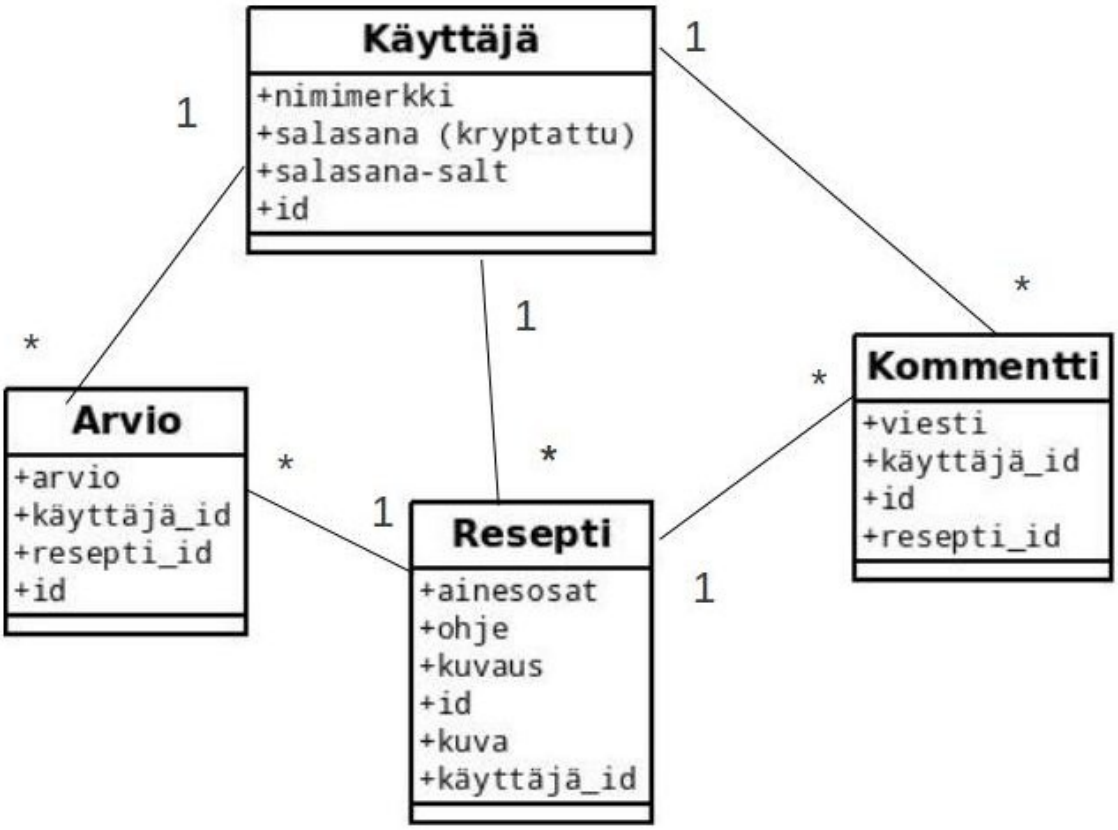
Johdanto

Tavoitteenani oli tälle projektille koodata yksinkertainen web-keittokirja Ruby on Rails -alustalle. Järjestelmän tarkemmat tavoitteet löytyvät aihekuvauksesta, mutta yksinkertaisuudessaan keittokirjassa voi lisätä, selata ja kommentoida reseptejä.

Kuten edellä mainitsin, olen ohjelmoinut sovelluksen ruby on railsilla. Selaimelta tämä ei vaadi mitään erikoista, mutta palvelimen täytyy tukea Rubya. Tästä syystä en voinut pistää sovellustani yliopiston palvelimille, vaan se on testattavana herokuapp.com -sivustolla (osoitteessa “keittokirja.herokuapp.com”)

Käytin projektin alussa Railsin oletustietokantaa SQLite, mutta jouduin herokua viritellessäni vaihtamaan sen postgresiksi. Vaihto onnistui, mutta jonkin verran säätämistä ja asiaan perehtymistä. Vaikeudet johtuivat ehkä postgresin monipuolisuudesta ja sitä myötä monimutkaisuudesta SQLiteen verrattuna.



Käyttäjäryhmät

Kirjautumaton käyttäjä

kirjautumaton on kuka tahansa sivustoa selaava henkilö

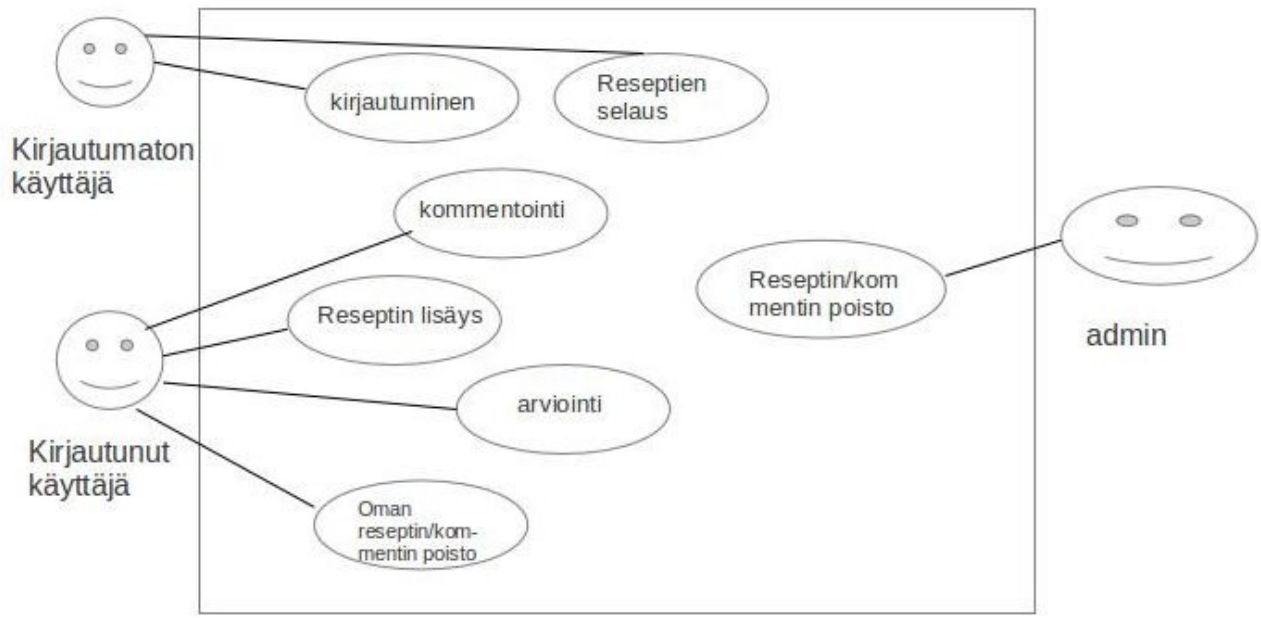
Kirjautunut käyttäjä

Kirjautuneella käyttäjällä on tunnus järjestelmässä ja hän on kirjautunut sisään.

Admin

Kirjautunut käyttäjä, jolla on attribuutti Admin = true. Adminiuden voi asettaa vain konsolista käsin.

Käyttötapaukset



Jokainen käyttäjäluokka perii myös alempien käyttäjäluokkien käyttötapaukset.

Kirjautumattoman käyttäjän käyttötapaukset

-Reseptien haku & selaus

Käyttäjä voi hakea eri hakusanoilla reseptejä ja tarkastella niitä lähemmin. Kirjautumaton käyttäjä näkee kommentit ja arviot, mutta ei voi itse kommentoida tai arvioida reseptejä.

-Tunnuksen luominen

Käyttäjä syöttää haluamansa nimimerkin, salasanan ja salasanan varmistuksen. Jos käyttäjätunnus on vapaa ja salasana riittävän pitkä ja yhtäläinen varmistuksen kanssa, tunnus on luotu onnistuneesti, käyttäjä siirtyy etusivulle ja tunnuksen luomisesta tulee ilmoitus sivun ylälaitaan.

-Kirjautuminen

käyttäjä syöttää aiemmin luodun tunnuksen ja salasanansa ja vaihtaa tilansa kirjautuneeksi käyttäjäksi, jos käyttäjätunnus ja salasana täsmäävät.

Kirjautuneen käyttäjän toiminnot

-Reseptin lisäys

Käyttäjä voi lisätä uudedn reseptin kaavakkeella. Jos resepti täyttää kaikki vaatimukset (ei puuttuvia kenttiä), resepti tallennetaan tietokantaan. Jos ei, käyttäjälle tulee siitä ilmoitus ja käyttäjä palaa reseptin luontisivulle.

-Reseptin kommentointi

Käyttäjä voi lisätä mihin tahansa reseptiin lyhyen kommentin.

-Reseptin arviointi

Käyttäjä voi arvioida muun käyttäjän luoman reseptin. Reseptejä arvioidaan ennalta määrätyllä skaalalla. Kukin käyttäjä voi arvioida yhden reseptin vain kerran, mutta arvosanaansa voi muuttaa.

-Reseptin poisto

käyttäjä voi poistaa itse lisäämiään reseptejä.

-Reseptin muokkaus

Käyttäjä voi muokata omia reseptejään, jolloin näkymä ohjataan muokkaussivulle. Kaiken julkisen informaation reseptistä voi muokata

-Uloskirjautuminen

painamalla kirjaudu ulos -nappia käyttäjä vaihtaa tilansa kirjautumattomaksi käyttäjäksi.

Järjestelmänvalvojan toiminnot

-Reseptin tai kommentin poisto ja muokkaus

järjestelmänvalvoja voi poistaa kommentin tai reseptin tai muokata reseptiä ikään kuin se olisi hänen oma reseptinsä.

Järjestelmän yleisrakenne

Sovellus on järjestetty käyttäen Ruby on railsin yleisiä käytäntöjä, eli MVC-mallin mukaisesti. /app -kansiossa on kansiot model, view ja controller, jotka sisältävät nimiensä mukaiset tiedostot.

Model on yhden tietokannan tietotyypin kuvaus. Resepteille, käyttäjille ja kommenteille on kaikille oma malli.

Tärkein sovelluslogiikka on kontrollereilla. Kontrollerit vastaanottavat näkymiltä tietoa, ohjaavat sitä malleille tallennettavaksi ja ohjaavat taas käyttäjän oikeaan näkymään. Kontrollerit (ja mallit) on puhdasta Rubya. Jokaiselle eri tietosisällölle on oma kontrollerinsa. Kontrollerin metodit ovat käyttäjän tietosisällöille tekemiä toimintoja, kuten selaus, lisäys ja muokkaus.

Kontrolleri application_controller ei sisällä minkään tietosisällön metodeita, vaan sisältää kokoelman ohjelman tarvitsevia apumetodeja. Metodeja voidaan kutsua sekä näkymistä, että muista kontrollereista. Apumetodien käyttötarkoitus on kommentoitu koodiin.

Useimmille kontrollerin metodille on oma näkymänsä. Metodiin liittyvä näkymä on oletus. Siirtymä muuhun kuin olteusnäkymään on täytynyt koodata erikseen metodiin. Näkymä application.html.erb on poikkeus. Siinä määritellään sivuston yksittäisestä näkymästä riippumattomat komponentit, kuten ylärivin banneri ja linkkipalkit.

Näkymät (views) vastaavat yhtä html-sivua. Näkymät eivät tee juurikaan itse mitään logiikkaan liittyviä asioita, vaan vastaanottavat ja lähettävät kontrollereille käyttäjän toimintoja. Näkymät on kirjoitettu Embedded Rubyna, eli html:nä johon on upotettu “<%” -tagien sisään Ruby-koodia toiminnallisuuden mahdollistamiseksi.

Järjestelmän adminius on toteutettu hyvin yksinkertaisesti käyttäjään liittyvällä boolean-arvolla. Adminin ominaisuus on, että hän voi poistaa kaikkia reseptejä ja kommentteja niin kuin ne olisivat hänen omiaan (omistamisen aputarkistava metodi hyväksyy aina adminin omistajaksi).

Järjestelmän komponentit

Kirjastot

Sovellukseni käyttää Ruby on Railsille tyypillisesti useita kirjastoja, “gemejä”, jotka on määritelty suoraan juurihakemistossa olevassa Gemfilessä. En erittele kaikkia gemejä, sillä suurin osa kuuluu rails-projektiin oletuksena. Selostan kuitenkin itse lisäämäni merkittävät kirjastot.

1. **Paperclip ja ImageMagick**

kuvien lisäämistä ja näyttämistä tietokannassa helpottaa Paperclip. Paperclipin avulla pystyn helposti tallentamaan ja liittämään html:n käyttäjän lomakkeella syöttämän kuvan. Kuvaa pystyy muokkaamaan kevyesti paperclipin avulla käyttäen ImageMagick -ohjelmaa. Liitännäinen ImageMagickille on gem 'rmagick'. Rmagickin lisäksi palvelimella tulee olla varsinainen imageMagick asennettuna.

Sovellus käyttää imageMagickin monipuolisia muokkaustoimintoja vain kuvan koon määrittämiseen. Kuvasta tallennetaan sekä pieni (100x100px) versio selaussivua varten, että suuri kuva (max 600x600px) reseptin omaa sivua varten.

2. **BCryptRuby**

Salasanojen tallentamiseen salattuna käytän Bcryptia, sillä se on ilmeisesti suositeltu kirjasto tähän tarkoitukseen. Kun käyttäjä syöttää valitsemansa salasanan, sitä ei tallenneta selkokielisenä tietokantaan (tämä antaisi mahdolliselle tietokantaan murtautujalle suoraan käsiinsä kaikkien käyttäjien salasanat) vaan salasana tallennetaan kryptattyna hashina (eli sovelluskaan itse ei tiedä oikeita salasanoja). Kun käyttäjä kirjautuu, salasanasta muodustetaan taas hash, ja saatua hashia verrataan tallennettuun hashiin. Näin järjestelmään voi kirjautua vain oikealla salasanalla, jonka tietää vain alkuperäinen käyttäjä.

Tässä on toki ongelma, että mahdollinen murtautuja voisi olla generoinut alunperin valtavan tietokannan hasheja mapattuna ne generoiviin salasanoihin (käyttäen Bcryptin hash-funktiota). Tästä syystä jokaiselle käyttäjälle generoidaan luonnin yhteydessä satunnainen “salt” -merkkijono. Aina ennen kryptausta salasanaan lisätään tämä “salt”.

Saltin lisääminen tekee sen, että murtautujalla, joka käyttää tietokantaa, tulisi olla erillinen tietokanta jokaiselle mahdolliselle saltin arvolle. Saltin lisäys tekee siis tietokannan käyttämisestä tehottoman konstin murtaa kryptatut salasanat.

Bcryptin ominaisuuksiin kuuluu myös hitaus. Salausfunktiossa hitaus on hyvä asia, sillä se tekee kaikkien salasanojen kokeilusta hyvin hitaan, jopa mahdottoman hitaan ollakseen hyödyllinen.

Asennustiedot

Sovellus käyttää Railsin versiota 4.0 ja Rubya 2.0. Lisäksi sovellus vaatii ImageMagickin asentamisen.

Käynnistys/Käyttöohje

Sovellus on testattavissa osoitteessa **keittokirja.herokuapp.com** (huom, ei www:tä eteen!) Herokuapilla on hieman ongelmia kuvien kanssa, mutta muuten sovelluksen pitäisi vastata omalla koneella kehittämääni.

Testaus, tunnetut bugit ja jatkokehitysideat

Olen testannut ohjelmaa vain käsin, mutta huolellisesti, kehittäessäni ohjelmaa. Pyöritin sovellusta aluksi koodatessa omalla koneellani, mutta myöhemmin laitoin sovelluksen herokuun, ja olen pystynyt testaamaan sitäkin kautta.

Ohjelman herokuversiossa on jotain ongelmaa kuvien kanssa. Heti tietokantaan tallennuksen jälkeen reseptin kuvat näkyvät normaalisti, mutta pienen ajan päästä osa kuvista häviää, tai ainakin pienenee huomattavasti. Pieneneminen ei nyt ole paha vika, mutta kuvien häviäminen on huolestuttavampaa. Vika saattaisi olla herokun palvelimen ja imageMagickin yhteistyössä.

Jatkokehitystä miettien, ainakin reseptien hakua voisi kehittää. Alunperin ajattelin lisätä mahdollisuuden laittaa reseptiin tageja hakusanoiksi. Resepteillä voisi olla myös paljon muita attribuutteja, joiden avulla niitä olisi vielä helpompi hakea.

Tällä hetkellä reseptien etusivu näyttää kaikki tietokannassa olevat reseptit. Pienellä aineistolla se toimii, mutta jos reseptejä olisi enemmän, olisi turha pitää ne kaikki samalla sivulla. Reseptien järjestämistä etusivulla voisi myös pohtia.

Ohjelmassa on paljon yksityiskohtia, joita viilaamalla siitä saisi paremman.

Omat kokemukset

Ohjelman koodaaminen oli suurimmilta osin hauskaa. Rubya on mukava koodata, ja Rails on melko looginen järjestelmä, johon on helppo päästä sisälle. Toki Apia, StackOverFlowta ja RoR-tutoriaaleja tuli selattua paljon, mutta se kuuluu asiaan. Ongelmat harvemmin olivat kovin pitkäkestoisia kun ne liittyivät suoraan ruby on railsin koodaamiseen.

Suurimmat haasteet ja ärsyttävimmät bugit oli muiden ohjelmien käytössä railsin yhteydessä. Imagemagickin ja Paperclipin yhteistyö vaati paljon säätämistä, vaikka varsinainen ohjelman käyttö olikin todella helppoa. Laittaessani sovelluksen herokun serverille jouduin vaihtamaan SQLiten postgresiin, mikä vei myös paljon aikaa.

Jos tekisin nyt uuden harjoitustyön, tekisin sen ehkä Railsin sijaan php:lla. Olisi ollut hyödyllisempää opetella websovelluksien ohjelmointia aloittaen matalammalta tasolta, jolloin periaatteet tulisivat selvemmiksi. Toki tässäkin projektissa opin paljon web-sovelluksien ja html:n koodaamisesta, mutta RoR piilottaa useita asioita, joita aloittelijan olisi hyvä tietää.