

## Lyhyesti

Tässä siis www-sovellusten harjoitustyön dokumentaatio, jossa esitellään valittu harjoitustyön aihe, toteutetut ominaisuudet suhteessa harjoitustyön vaatimuksiin, työn toteutustapa ja teknologiat, ohjeet virittämisen ajamiseen sekä lyhyesti käyttöohjeet itse toteutetulle systeemille. Tämä on siis harjoitustyön dokumentaatio, kun ei erikseen vaadittu jotakin tietäntyyppistä dokumentaatiota kirjoittamaan, niin valitsin tyylin itse.

## Aihe ja ominaisuudet

Harjoitustyön aiheeksi valitsin kuvienjakopalvelun, johon käyttäjät voivat rekisteröityä ja johon he voivat ladata omia kuviaan ja jossa he voivat selata ja kommentoida toisten kuvia. Itse sovellus/palvelu voi olla jotenkin ankea noin käytettävyyden puolesta ja rajoittunut pelkästään kuvien lataukseen ja kommentointiin, mutta ajaa periaatteessa asiansa harjoitustyönä, kuvittelisin. Vähän harmittaa kun en enempää ehtinyt tehdä.

Alla taulukoituna harjoitustyön vaatimukset ja vastaavasti toteutetun työn ominaisuudet, taulukko käytännössä kopioitu kurssisivulta ja hieman viilattu:

Ominaisuus	Pisteet	Toteutettu	Kommentit
Responsiivinen ulkoasu	5	✓	ulkoasu skaalautuu/mukautuu näytön koon mukaan, käytetty @media -toimintoa CSS:ssä
Tietokannan käyttö tietojen tallennukseen	5	✓	SQLite 3 (Pythonin sqlite3-kirjastolla)
Käyttäjän autentikointi (luonnollisesti hashit ja suolat mukana)	5	✓	modules/database.py
Roolipohjainen käyttäjänhallinta (vähintään pääkäyttäjä, peruskäyttäjä ja vieras)	+2	✓	data/database.sql → usergroups-taulu, yleisellä guest-käyttäjällä kirjautuminen toimii syöttämällä käyttäjätunnukseksi 'guest' eikä salasanaa tarvita
Front-controllerin käyttö	3		
MVC-mallin mukainen sivusto	5	✓	Tavallaan kyllä, data tallennettu levyille (model), käyttäjälle esitellään datasta generoituja html-näkymiä (view), käyttäjä syöttää käskyjä sovellukselle joka manipuloi dataa (controller) – tavallaan siis MVC-malli
Välimuistin käyttö	3		
SEO (hakukoneoptimointi)	3		
Kolmannen osapuolen palvelun käyttö	5		

(esim. FB-login, [GA ei riitä])			
Toinen kolmannen osapuolen palvelun käyttö (säätiedot, smartpostit, elokuvatarjonta...)	3		
Yksikkötestit olemassa	5		
JSONin (/XML:n) käyttö tiedon liikuttelussa/tallennuksessa	3	✓	kommenttien lisäyksen ja latauksen toteutus ajaxilla käyttää JSON-muotoa datan siirtoon
jQueryn / Reactin / AngularJS:n käyttö	3	✓	jQuery ahkerassa käytössä
Dynaamisen SVG-grafiikan käyttö	3		
Canvas-elementin käyttö	3		
Sivuston sisältö tulostuu paperille luettavassa muodossa (aka on erillinen printtityyli)	2	✓	toteutettu CSS:ssä @media-direktiivillä, static/style/print.css
Sivusto ei sisällä yhtään kuvaa, vaan kaikki grafiikka on tehty dynaamisesti CSS:llä ja muilla tekniikoilla. Poislukien esimerkiksi tuotekuvat web-kauppasivustolla, sosiaalisen median profiilikuvat ym.	1	✓	ainoastaan käyttäjien lataamat kuvat ovat kuvia, muu grafiikka tehty CSS:llä (esim. liukuvärit, varjot, jne.)
Ajaj (/ajax) -ohjelmoinnin hyödyntäminen	3	✓	kommenttien lataus ja lisäys kuviin toteutettu ajaxilla
Sivusto tarjoaa sisältöä ladattavaksi PDF-muodossa (eli sisällöstä generoidaan PDF)	2		
Kattava dokumentaatio	5	✓	tässä tämä dokumentti
Hyvä vertaisarviointi (1pts. per arviointikategoria ja +1pts erinomaisesta arviosta)	5		vaikea sanoa etukäteen
Esteettömyys (kts. W3C:n <a href="#">määritelmä</a> ja <a href="#">tarkastusohjeet</a> )	3		ei ole aikaa toteuttaa tuota kaikkea...
Implementoi joku standardi (järkevä) kolmannen osapuolen yleisesti käytetty komponentti (latauspalkki, datetimepicker, dialogi, ...) käsin	3		
Selaimen yhteensopivuustarkastus (tukeeko selain esim. käyttämäsi canvasta? Jos ei, niin ilmoita)	2		
Animoidut transiitiot (uusi elementti ilmestyy, joku elementti muuttuu jne., niin käytetään animaatioita tai efektejä)	2	✓	toteutettu jQuery:n efekteillä, esimerkiksi kirjautumisessa ja käyttäjänluonnissa viestit häipyvät jQueryn fadeOut-efektillä
(Lähes) reaaliaikainen kommunikointi AJAX:lla esim. lentely- tai toimintapeli-tyyliseen projektiin tai chat-tyyliseen projektiin (WebSocket +3 pts päälle, mutta en lähtisi WebSoketteja aivan kylmiltään tekemään)	3	✓	kuvien kommentointi tapahtuu käytännössä lähes reaaliajassa ajaxilla, tällä hetkellä koodi lataa kommentit uudestaan viiden sekunnin välein (voisi olla lyhyempikin tietysti se väli)
JavaScriptissa käytetty 'use strict' -direktiiviä	-0	✓	toteutettu kaikissa omissa skripteissä

CSS:ssä ja JavaScriptissä ei ole ylikäytetty #id-selektoria	-0	✓	id-delektoria käytetty vain kahdesti tai jotakin, muuten käytetty luokkia ja luovaa elementtien järjestelyä
koodi on kirjoitettu englanniksi ja kommentoitu kunnolla	-0	✓	en ole aivan kaikkea itsestäänselvyyksiä kirjoittanut kommentteihin, mutta muuten on kommentoitu mielestäni kattavasti

## Arvosanatavoite

Edellä esitellyssä taulukossa eriteltyjen toteutettujen ja toteuttamattomien ominaisuuksien ja niistä annettavien pisteiden perusteella pistemääräksi muodostuisi niiden vaatimusten perusteella (poislukien vertaisarvioinnin pisteet ja muu vastaava mitä ei itse tällä hetkellä kykene arvioimaan) seuraavaa, huomioiden arviointiperusteet ja muu selostus kurssisivulla:

mikä	miten paljon	ajatuksia
ominaisuuksien perusteella	44	taulukko yllä
pyöristettynä ohjeiden mukaan	40	alaspäin pyöristys
toteutuksen laatu huomioitu	40	toteutus ei omasta mielestäni ole mitenkään purkkaratkaisu, joten en ainakaan itse vähentäisi pisteitä kovinkaa paljoa... ??

## Toteutustapa ja teknologiat

Työ on toteutettu käyttäen seuraavia tekniikoita/teknologioita, kun ei erikseen ollut tehtävänannossa rajattu esimerkiksi ohjelmointikieltä:

- Python 3 -ohjelmointikieli
- sqlite3-, uuid-, os- ja muut tarpeelliset kirjastot Pythonissa sisäänrakennettuna
- Flask-mikroframeworkki Pythonille palvelinpuolen toteutukseen (html-templaattit, http protokollalla-kommunikointi)
- Passlib-kirjasto Pythonille salasanan häshäämiseen ja suolaukseen
- verkkopalvelinohjelmana käytetty Flaskin sisäistä palvelinta joka on tosin suunnattu vain kehitysvaiheen käyttöön mutta riittää tällä kertaa
- käyttäjäpuolella HTML, CSS, JavaScript, jQuery

Datan varastointi, datan manipulointi ja käyttöliittymä on tavallaan erotettu toisistaan. Data tallennetaan SQLite3-tietokantaan palvelimella, kuvat tallennetaan levyille. Käyttäjälle näytetään verkkosivuja joihin on täytetty dataa ja joiden kautta käyttäjä välittää käskyjä palvelimelle sovellukselle, joka muokkaa dataa syötteen perusteella.

## Ympäristö

Ympäristönä käytetty Windowsia ja Linuxia (Ubuntu) ristiin rastiin kehityksessä, pitäisi toimia molemmilla. Vaatii Pythoniin oleelliset kirjastot. Python 3.6.x riittää, lisäksi kirjastot Flask (sekä edelleen sen vaatimukset) ja Passlib. Seuraava komento käyttää pythonin pip-palasta asentaakseen tarvittavat kirjastot nykyiselle tietokoneen kirjautuneelle käyttäjälle (ei siis koko koneen saataville):

```
python3 -m pip install --user flask passlib
```

Missä *python3* pitää korvata pelkällä *python* jos Windowsilla on pelkkä python.exe jossakin. Linuxilla pitää olla python3 koska muuten käyttää Python 2:ta. Koodi on tallennettu Githubiin, josta sen saa komennolla:

```
git clone https://github.com/Contrathetix/WWW-Course-Assignment
cd WWW-Course-Assignment
```

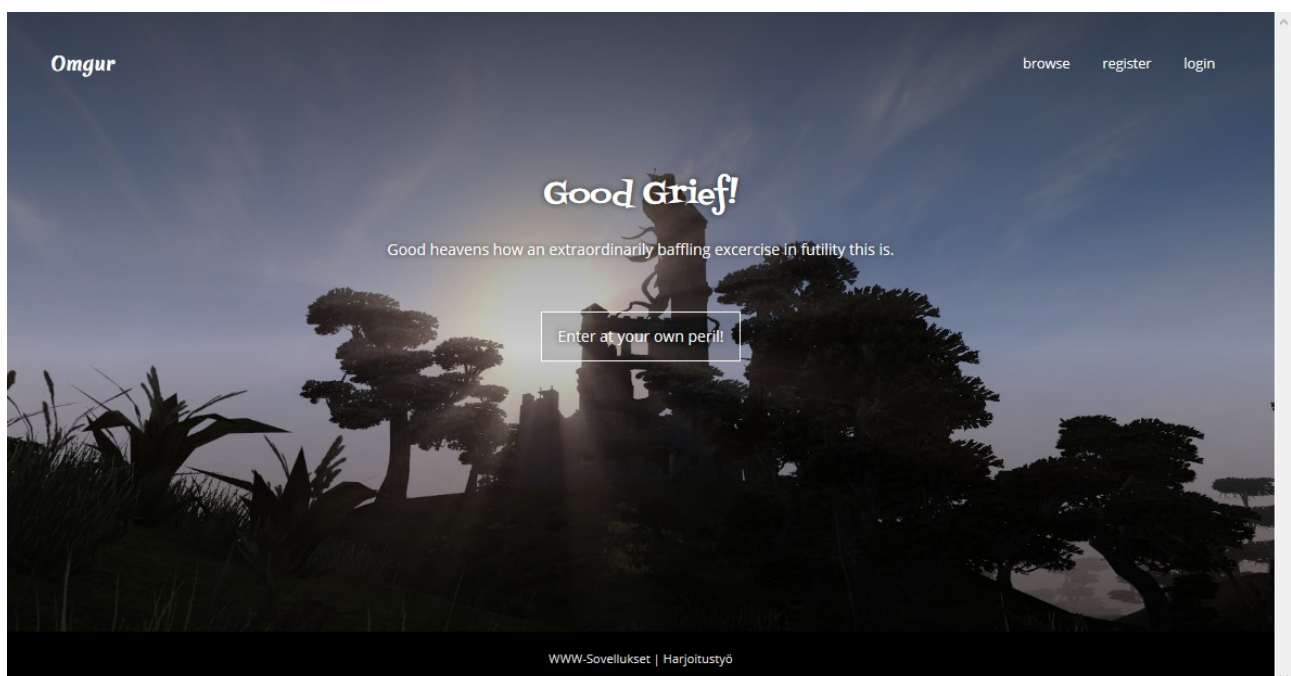
Ajaminen tapahtuu pythonilla, missä taas pitää Windowsilla korvata *python3* pelkällä *python*:

```
python3 app.py
```

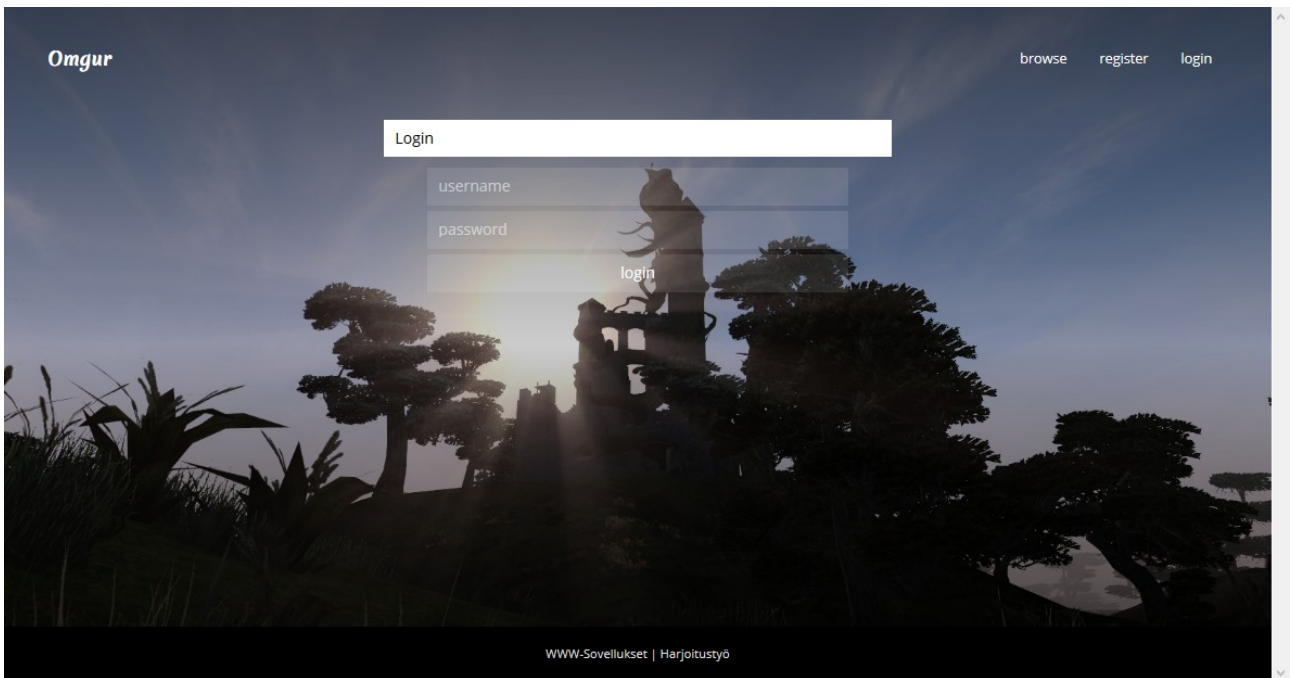
Ohjelma pyörii osoitteessa "http://localhost:8080" jos kaikki toimii kuten pitää.

## Käyttöohjeet

Lyhyesti vielä dokumentaatioon käyttöohjeista, sen jälkeen kun ohjelma on käynnistetty. Ohjelma resatoi käynnistyessään tietokannan ja kumittaa kaikki ladatut kuvat. Oletuksena lisätään vain guest-käyttäjä joka ei tarvitse salasanaa kirjautuakseen. Aloitussivuna on tyyllitelty sivu, josta pääsee linkkejä painamalla joko selaamaan kuvia, kirjautuman tai rekisteröitymään. Kirjautuneet käyttäjät voivat kirjautua ulos, selata kuvia tai ladata omia kuvia.



Kirjautumisvaiheessa tarvitsee syöttää tunnut ja salasana. Salasana on hashattu ja suolattu. Vieraskäyttäjälle “guest” pääsee ilman salasanaa.



Omgur

browse register login

Login

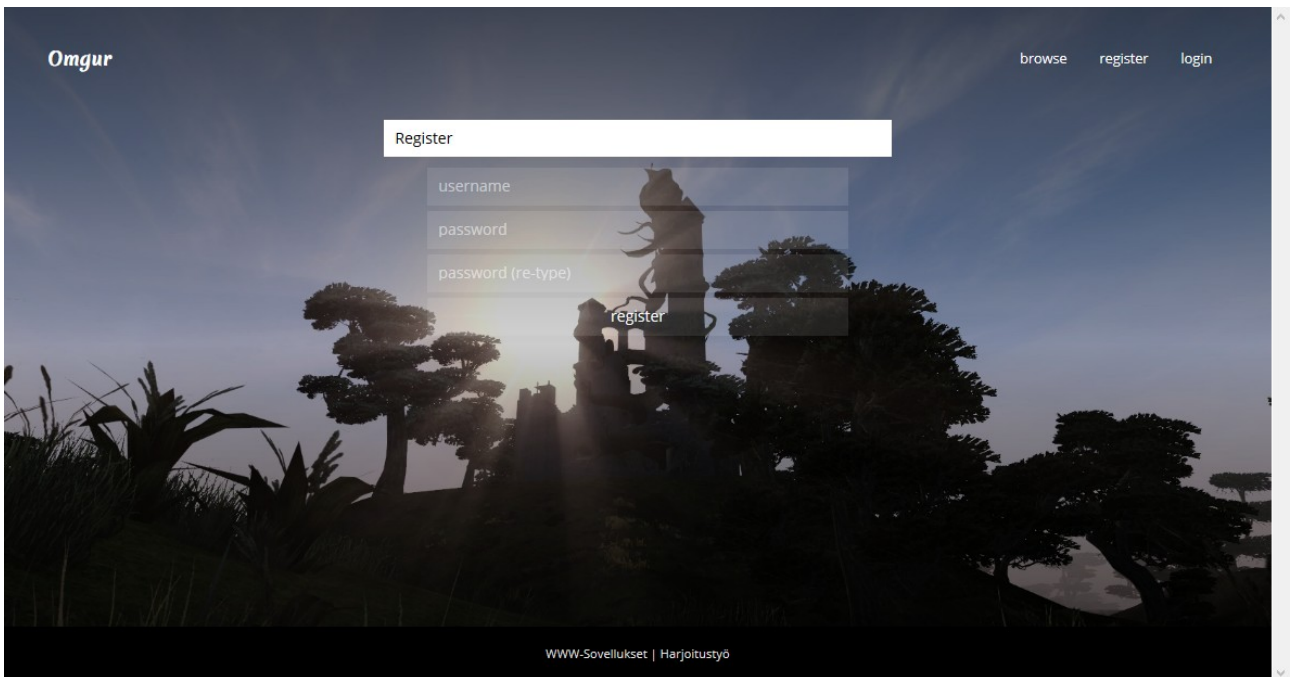
username

password

login

WWW-Sovellukset | Harjoitustyö

Rekisteröityessä tunnus ja kaksi kertaa salasana syötettävä.



Omgur

browse register login

Register

username

password

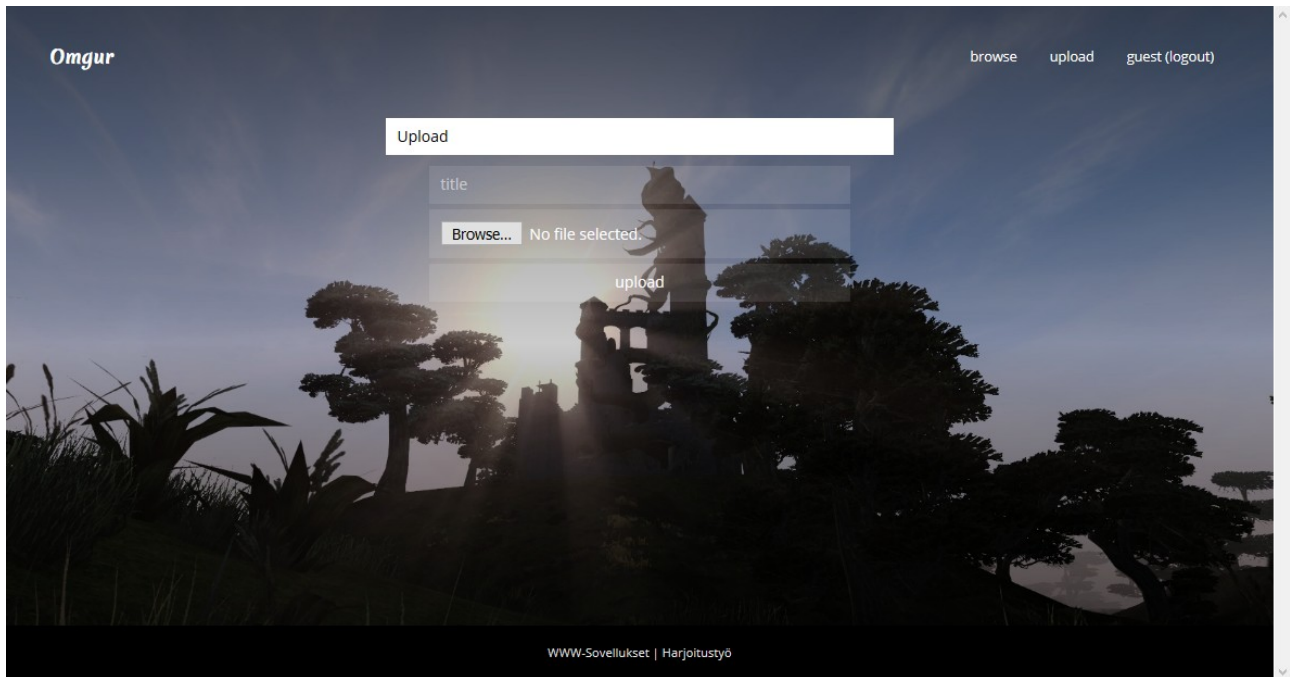
password (re-type)

register

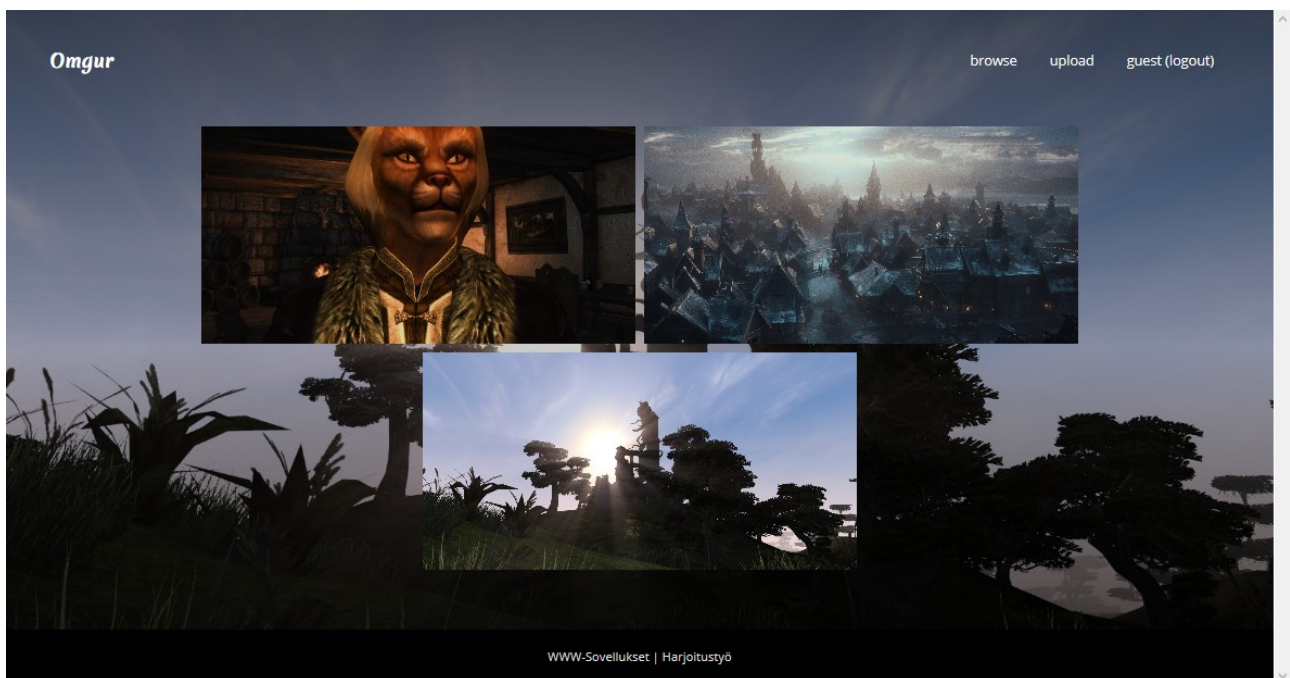
WWW-Sovellukset | Harjoitustyö



Kuvaa ladattaessa tulee antaa itse kuvatiedosto ja otsikko.



Kuvia voi selata vapaasti myös kirjautumatta.



Kommentteja voi lisätä kuviin jos on kirjautunut sisään, kommentit päivittyvät automaattisesti tietyn ajan välein, jolloin kommenttiosio voinee toimia myös eräänlaisena chattina (joskin keskustelut olisi kuva pitää lyhyinä ja asiallisina, kun ne jäävät kaikkien näkyville).

