BACK-END DETAIL DESIGN

Cheery

Description

This Detail Design will give in depth information about the back-end of this project.

Melle Dijkstra
Melle210202@gmail.com
8-12-2015

1 Table of contents

2 Server Initialization

2.1 Cloud Server (droplet)

To put it simply, a cloud server is a virtual server which run on cloud computing environment. It's just like an actual server but it's virtual which has its advantages. You can manage hardware and software really easy. Upgrading goes within minutes. Also with a cloud server you can manage your own software that is run on it. The only disadvantage is that you can't physically touch and change parts of the server.

The cloud server is bought at Digital Ocean (https://www.digitalocean.com/) for $5.00 per month. Digital Ocean is a professional cloud server service for high quality servers which they call droplets. Also Digital Ocean has an API which makes is better but is not relative for this project.

Statistics of the cloud server (droplet) when bought:

- IP Address: 178.62.143.203
- Gateway: 178.62.128.1
- Netmask: 255.255.192.0
- Memory: 512MB (RAM)
- Disk: 20GB (Storage)
- Region of server: AMS2 (Amsterdam 2)
- OS: Ubuntu 14.04 x32 (vmlinuz-3.13.0-57-generic)

The droplet is installed with a 32 bit OS because it only has 512MB RAM which is not much and by choosing 32 bit OS it performs better.

2.2 Server Configuration

The configuration all needs to be done manual because that's the way Digital Ocean works. Here is a list of all the programs needed for the project and their configuration:

2.2.1 Operating System

The Operating System that will be used is Ubuntu 14.04 x32 (Server version). This is the latest stable version of Ubuntu which is always a good pick. The Ubuntu distribution is chosen because is used by a large amount of people which helps if there are any problems.

The 32 bit version is used because the RAM of the server is only 512MB which goes better with a 32 bit version. This won't be a problem for the programs that are used.

The Ubuntu server edition is used for this project because it's lightweight. It's lightweight because there isn't a fancy user interface (front-end). This means not a lot of space is used for the Operating System and the server will run smoother.

2.2.2 SSH (Secure Shell)

SSH will be used to configure the server from any location. SSH is secured with user and password registration. The root account (which has all privileges) will only be used to configure the server. Next to root there will be a web-admin user account which will have access to the files of the program and not those on the server. This is to prevent any hackers or malicious users to change files that aren't originated of the project. The web-admin account is able to use the basic CRUD operations on the project files.

Also for security reasons the SSH port is changed from the default 22 port to 22222.

2.2.3 Firewall

The firewall will be configured with the program "ufw". Ufw stands for Uncomplicated Firewall and is used to configure the firewall in an uncomplicated way. The commands are easy to execute and setting a secure firewall up is done in minutes. The configuration for the firewall is described below:

Rule	Port	Reason
Allow	80	This port is the standard web port where the HTTP layer communicates. This port has to be open to allow communication between the website and the users connecting with it.
Allow	22222	The default SSH port (22) is changed to 22222 for security reasons. This port allows administrators to login with SSH and configure the server.
Deny	all	Deny all incoming communications on all other ports than above for security reasons because other communications is not needed.

2.2.4 Apache Webserver

When Apache is installed there isn't a lot you have to do make it work. The only thing is start the Apache service and it runs! The version installed is Apache 2.4.7 (Ubuntu) and PHP version 5.5.9.

But because for this project there is an API and a development environment we need to make 2 different virtual hosts. These are the steps to make a virtual host:

1) Create a directory on the server what will be the document root of the virtual host (example api.cheery.nl goes to /var/www/api, see Directory & File Structure).
2) Grant permission to apache to use the directory (chown –R $USER:$USER /folder/location) and set permissions so everyone can see the page (chmod –R 755 /folder/location)
3) Create a new virtual host file (nano /etc/apache2/sites-available/api.cheery.nl.conf) be sure it has the .conf extension and has the configuration it needs
4) Enable the virtual host by running "a2ensite api.cheery.nl.conf"
5) Check if you can connect to api.cheery.nl and see the directory index from /var/www/api

2.2.5 MySQL Database

For this project MySQL is used to store all permanent information like users, groups, notifications and many more. version installed is MySQL 5.5.44.

MySQL is a powerful database management system used for organizing and retrieving data. MySQL is the most popular database system used with PHP. PHP has a library which can make MySQL connection and send queries it.

MySQL is setup to use a username and password before any operations can be executed. That's why only the back-end can talk to the database and the information retrieved is used to display the page.

To manage the database the command line can be used to directly talk to the database. But for ease of access phpMyAdmin is used to manage the information in the Cheery database. The phpMyAdmin directory is secured with a password.

2.2.5.1 Tables

In MySQL the data is stored in tables with fields. Here are the tables used for the Cheery project. **The tables and the whole database can be changes throughout the project development.**

Users table
This is the table where everything about the users is stored. This information is used to display profile pages and authenticate which visitor is which user.

Column	Type	Default	Null	Comments
user_id	int(11)		No	The ID to identify a user
firstname	varchar(50)		No	The first name of the user
lastname	varchar(50)		No	The last name of the user
email	varchar(200)		No	The email of the user used to login
password	varchar(64)		No	The password to authenticate a user combined with his email
avatar	varchar(200)	NULL	Yes	The profile image location of the user
cover	varchar(200)	NULL	Yes	The cover location of the user
points	int(11)	20	No	The points of the user
gender	char(1)	NULL	Yes	The gender of the user
birthday	date	NULL	Yes	The birthday of the user. Used to generate rewards for the user on his birthday (optional)
developer	tinyint(1)	0	No	Check if a user is a developer
website	varchar(100)	NULL	Yes	The website of the user. Used on the profile page to link to his webpage (optional)
country	varchar(50)	NULL	Yes	The country where the user lives in (optional)
nickname	varchar(20)	NULL	Yes	A nickname of the user. It is used in the search engine and for dynamic URL's
mobilenumber	varchar(25)	NULL	Yes	The mobile number of the user (optional)

Column	Type	Default	Null	Comments
lastactivity	int(11)	NULL	Yes	The last time the user was active in a UNIX timestamp
created_at	timestamp	CURRENT_TIMESTAMP	Yes	The date + time when the user created his profile
api_key	varchar(32)	NULL	Yes	The API key for the user so he can use the API from Cheery
rmID	varchar(128)	NULL	Yes	The remember ID used to check the remember cookie
rmToken	varchar(128)	NULL	Yes	The remember token which is hashed and needs to be checked with the non-hashed credentials from the user

Groups table
The groups table stores all the information about the groups. Cheery uses the information to display group information on the webpages.

Column	Type	Null	Default	Comments
group_id	int(11)	No		The ID to identify the group
name	varchar(20)	No		The name of the group
owner_id	int(11)	Yes	NULL	The creator his ID. Used to check which group is from which user
avatar	varchar(200)	Yes	NULL	The groups image/avatar location
cover	varchar(200)	Yes	NULL	The location of the group's image
color	varchar(6)	Yes	NULL	The color from the group in HEX value
points	int(11)	No	0	The points from the group
type	char(1)	No	g	The type of the group g=group, business=b, friends=f, country=c
created_at	timestamp	Yes	CURRENT_TIMESTAMP	The timestamp when the group was created

Notifications table
The notifications table stores all the information about notifications send to different users. These are used to update the user about specific events.

Column	Type	Null	Default	Comments
notification_id	int(11)	No		The ID of the notification
for_id	int(11)	No		The user_id for which the notification is for
notification	varchar(500)	No		The notification text
type	int(11)	No		The type of notification in numeric codes
has_read	tinyint(1)	No	0	Check if the user has read the notification
created_at	timestamp	No	CURRENT_TIMESTAMP	The timestamp when the notification was created

FriendRelations table
The FriendRelations table stores all the information about users who are friends with other friends and request send to each other.

Column	Type	Null	Default	Comments
first_user	int(11)	No		The ID of the first user in the friend relation
second_user	int(11)	No		The ID of the second user in the friend relation
Status	tinyint(1)	No	0	The status of the relation. 0 = not friends (request) 1 = friends
created_at	timestamp	No	CURRENT_TIMESTAMP	The time when the request was send

UserGroupRelations table
The UserGroupRelations table stores all the information about users that are members of groups.

Column	Type	Null	Default	Comments
user_group_relation_id	int(11)	No		The ID of the User -> Group relation
user_id	int(11)	No		The user ID who is in the group
group_id	int(11)	No		The group ID to identify which group the user is in
join_date	timestamp	No	CURRENT_TIMESTAMP	The date when the user joined

Messages table

The Messages table is a simple table to store messages send between 2 users.

Column	Type	Null	Default	Comments
message_id	int(11)	No		The ID of the message
from_id	int(11)	No		The user ID from who the message comes from
to_id	int(11)	No		The user ID to who the message is sent
message	int(11)	No		The actual message content
send_at	timestamp	No	CURRENT_TIMESTAMP	The time when the message was sent

Settings table

The settings table stores all the settings from a user like showing his profile image to the public or showing him in search results. The type of visitors to some settings can be shown are specified by a numeric value. So with the settings on 0 nothing can be shown. How higher the settings the more people can see the user his content.

Column	Type	Null	Default	Comments
user_id	int(11)	No		The ID of the user from who the settings are
showFriends	tinyint(1)	No	0	Check to which type of visitors friends can be shown
showAvatar	tinyint(1)	No	0	Check to which type of visitors the profile image can be shown
showCover	tinyint(1)	No	0	Check to which type of visitors the cover image can be shown
showPoints	tinyint(1)	No	0	Check to which type of visitors the points can be shown

3 Project Management

3.1 Directory & File Structure

A file structure is a way of organizing the files and folders in logical way. When you put every file in a single folder it's not really a clear overview. The development of the project can go slower when you're not keeping a good directory & file structure. Also when files interact with each other by example including a file or for getting information it is much better to have a clear and plain structure for your project.

Here is the file structure for the Cheery project with every directory explained. The name after the bullets are the actual name of the folder (**the file structure can change throughout development**). These files are located on the server at **/var/www/**.

- Round bullets are folders
- ❖ Diamond shaped bullets are files

The API folder is the directory for all api.cheery.nl requests

- api

  The v1 folder stands for version 1. **Throughout development new versions may be created those will have their own folder.** The name will be just like older versions but then incrementing the version number so after v1 there will be a folder called v2. This method is used because if there are applications that use version 1 and the API gets a different structure, weird output will be generated and the program that makes use of the API will not work correctly

  - v1

    The .htaccess file redirects every request to the index.php so that he can handle all the requests

    - ❖ .htaccess

    The main index file

    - ❖ index.php

Continued on the next page

The dev folder is the directory where all dev.cheery.nl requests are handled. This is the development environment

- dev

    The lib folder holds all front-end libraries (frameworks) like jquery and materialize

    - lib
        - jquery
        - materialdesignicons
        - materialize

    The resources folder includes all self-written resources like CSS, JS and images needed to display on the site (not avatars, covers from users and groups)

    - res
        - css
        - images
        - js

    The testing folder includes files for testing and developing the site like easy session destroying, hashing, API calls and little projects

    - testing
    - ❖ .htaccess
    - ❖ Index.php

The html folder is the production environment where everything will be stored when going live

- html

    The avatars folders contains all the profile images from users and groups

    - avatars
        - groups
        - users

    The covers folder contains all the cover photos displayed on user and group pages

    - covers
        - groups
        - users

The includes folder contains all files that need to be included like classes, scripts, frameworks and configuration files

- includes
    - classes

    The flight folder contains all the files & folders needed for the flight framework more info about it here

    - flight

    Configuration file that gets included on almost every page. It has functionality like auto loading classes in PHP, starting PHP sessions, setting PHP configuration and the option to set error display on or off

    - ❖ config.php

The pages folder contains all the pages that get included dynamically through PHP. These pages may be filled with different information based on the input from the user

- pages

    The templates folder contains all the little pieces that are needed on the pages in the directory above. These can be dynamically filled lists, menus or just pieces that repeat on every page an can be easily included instead of repeated on every page

    - templates

3.2 Flight Framework

Flight is a fast, simple, extensible framework for PHP. Flight enables you to quickly and easily build RESTful web applications.

For this project the Flight framework will be used to create understandable URL’s for visitors for example cheery.nl/user?id=7 will be cheery.nl/users/7.

To install the Flight framework there are 2 options. The first one is downloading the package from flightphp.com/install and extract it to your web directory which is in case of this project the includes folder (see Directory & File Structure). Then include or require the /flight/Flight.php in a PHP file that needs the flight framework.

Then make sure that every request can be handled by the Flight framework by sending all traffic to the file where the Flight framework operates by putting a .htaccess file in that directory with this inside:

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php [QSA,L]
```

Then when installed you have all the functionality of the Flight framework. Here is an overview of the Flight framework with links to the original documentation.

- Routing
- Extending
- Overriding
- Filtering
- Variables
- Views
- Error Handling
- Redirects
- Requests
- HTTP Caching
- JSON
- Configuration
- Framework Methods

4 Cheery API

The cheery API will allow other developers all around the world to use information from the Cheery database. The API is located at api.cheery.nl and gives back JSON as a response.

To make the API as abstract and friendly possible the Flight framework is used once again. This allows developers to get information from API with easy understandable URL's which is a must for a good API. Ones the API is tested and will be production ready, he will perhaps be registered by Mashape which is a place where multiple API's are shared across the world.

4.1 API Structure

The structure of the API is very important and it needs to be easy to understand. The API is located at api.cheery.nl and the version number comes after like so api.cheery.nl/v1/users. When there needs to be changed something to the API it can develop further with version 2 which will then be located at api.cheery.nl/v2 like so. The links used in the documentation are only applicable to version 1 of the API. Newer version can use another URL structure.

The API will handle a request and give back a response in JSON like so:

```
version: 1,
method: "get",
count: 1,
result: [
        {
                user_id: "1",
                firstname: "Melle",
                lastname: "Dijkstra",
                email: "melle210202@gmail.com",
                gender: "m",
                avatar: "joker.jpg",
                points: "11020",
                birthday: "1998-03-09",
                developer: "1",
                website: null,
                country: "Holland",
                nickname: "Cheery Developer",
                mobilenumber: "06-11666686",
                created_at: "2015-11-04 06:46:21"
        }
        ]
```

With this you can retrieve information out of the JSON object notation and use the information for another application.

Obviously all sensitive and secured data cannot be retrieved by the API.

4.1.1 User Related Requests

To get information about users the “/users/” subsection needs to be used. Like so “api.cheery.nl/v1/users/”. You can then give some more input to the API to get specific information you need.

For example you can specify which fields to return with the “?fields=user_id,firstname,lastname” parameter. Like so:

http://api.cheery.nl/v1/users?fields=user_id,firstname,lastname

You can also specify the column to order the results by and which way to sort like so:

http://api.cheery.nl/v1/users?order_by=points&sort=DESC

You can also search for users with the API by specifying it with query parameter “?q=search” like so and combine it with other parameters:

http://api.cheery.nl/v1/users?q=lo&order_by=points&sort=desc

4.1.2 Group Related Requests

Getting information about groups is just as easy as the users section. You just replace the /users/ with /groups/ and you are done.

Retrieving specific information by specifying the field names.

http://api.cheery.nl/v1/groups?fields=name

Retrieving information and sorting it descending by points

http://api.cheery.nl/v1/groups?order_by=points&sort=DESC

4.1.3 Improvement

In time of development of the Cheery project a lot will change on the API side of the application. There will be some differences compared to the examples given here because an API can always be improved and big companies today even struggle to get a good understandable API that works for everyone.

When upgrading to a newer version of the API it should be as easy as replacing the version number used currently in the application to the version number to which it should be upgraded.

5 List of terms and abbreviations

A list with terms and abbreviations is described here for the difficult words used in this document. The list is sorted alphabetically.

Term / Abbreviation	*Description*
Back-end	The back-end is the location or place where the front-end communicates with. So a client (front-end) requests resources or operations from the back-end. The back-end is in most occasions the server. Back-end can also be the underlying code that executes when an event occurs in the front-end.
API	An API is a set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.
Front-end	The Front-end is the viewable/usable part of a system. This part is directly accessed by the user and allows easy use of the application. The front-end communicates with the back-end.
CRUD	Create, Read, Update, Delete operations
SSH	Secure Shell (SSH), sometimes known as Secure Socket Shell, is a UNIX-based command interface and protocol for securely getting access to a remote computer. It is widely used by network administrators to control servers remotely.
UFW	Uncomplicated Firewall. A program to setup a firewall with easy understandable commands.
Apache	A very popular open source, Unix-based Web server from the Apache Software Foundation (www.apache.org). There are versions for all popular Unix flavors, as well as Windows, and it is considered the most widely used HTTP server on the Internet.
Operating System	The low-level software that supports a computer's basic functions, such as scheduling tasks and controlling peripherals. Most common Operating Systems are Windows and Unix (Linux).
MySQL	MySQL is an open source relational database management system. Information in a MySQL database is stored in the form of related tables. MySQL databases are typically used for web application development.
Virtual Host	Virtual hosting is a method for hosting multiple domain names (with separate handling of each name) on a single server (or pool of servers). This allows one server to share its resources, such as memory and processor cycles, without requiring all services provided to use the same host name. For example http://example.com can be on the same server as https://anotherexample.org