System Requirement Specification & Installation Instructions

Portfolio Mapping tool (PMT)

Version 3.0

January 2019

Prepared by

Spatial Development International

Contents

1. Introduction 1
2. Definitions, Acronyms and Abbreviations 1
3. System Overview 1
4. User Characteristics 2
5. System Interfaces 3
6. Constraints 3
7. Hardware Requirements 4
   Database Hardware Requirements 4
   Application Hardware Requirements 5
   API Hardware Requirements 5
8. Software Requirements 5
   Database Software Requirements 5
   Application Software Requirements 5
   API Software Requirements 5
9. Resource Capacity Requirements 6
   General Concepts 6
   Application & API Management 6
   Database Management 6
   Server Technologies 6
10. PMT Framework & Instance Installation 7
   Database Installation Instructions 7
      Automated Database Backups 8
      Database Export/Import Feature and Automated Email Support 8
   API Installation Instructions 10
      API Download and Build 10
      PM2 Installation and API Setup 11
      Useful pm2 Commands 12
   Application Installation Instructions 12
      Download and Install the Application 12
References 14

spatialdev

Portfolio Mapping Tool (PMT)

System Requirement Specification

1. Introduction

The purpose of this document is to describe and specify the system requirements for the implementation and installation of the PMT. The document will also establish initial training, capacity and system architecture requirements. This document is primarily intended to serve as a guideline and reference for client preparation for an onsite implementation of the system.

2. Definitions, Acronyms and Abbreviations

Table 1

Term	Definition
API	Application Program Interface.
IATI	International Aid Transparency Initiative [1]
basemap	Refers to a collection of GIS data and/or orthorectified imagery that form the background setting for a map. [2]
host	A host, reference to network hardware, is a computer or other device connected to a computer network. Within this document, it refers to a server. [4]
PMT	Portfolio Mapping Tool
PMT Framework	Refers to all the PMT components (database, API, application) working together as a holistic system.
RDBMS	Relational Database Management System

3. System Overview

The Portfolio Mapping Tool 3.0 (PMT) is a web-based mapping application intended to help visualize where development work is taking place. The features of this web application allow users to browse, map, add, edit, and share projects without the need for geographic information systems expertise. It is ultimately intended to help users improve project strategies and partnerships for greater impact in their work. Its features and functions have been designed to provide the following benefits:

- **Inform strategic and project management decisions.** The PMT can help inform decisions by allowing users to take geographic information into account, whether it is the location of markets, related projects and partners, travel time, annual precipitation, or maize crop yields.
- **Communicate programmatic projects to key stakeholders.** A key benefit to users of the PMT is to see the spatial layout of their projects relative to context. Users can add their projects to the PMT database and then visualize those projects in a variety of ways.
- **Understand how programmatic efforts relate to other projects as well as to useful agricultural information.** Users are able to browse and map other people's projects alongside their own

projects. This functionality provides the framework for multiple organizations to communicate vital strategic information together in a coordinated fashion.

The PMT is designed to be flexible and expandable, allowing users to develop unique ways of applying the tool to their work.

4. User Characteristics

There are four main groups of users which will use the system:

- System Administrators
- System Managers
- Data Editors
- Data Viewers

System Administrators

System Administrators are concerned with data integrity, system stability and application functionality. This group has the highest computer skill set and capable of supporting all aspects of a system's hardware and software needs. Their interaction with the system is limited and necessary only for backing up and archiving the database, making application configuration adjustments, rebuilding the application after configuration changes, data management or maintenance made directly to the database, and to provide basic user support for other system users.

System Managers

System Managers interact with the system to manage user access, maintain high level data elements and oversee data management activities. These users have basic computer experience, have detailed understanding source data collection and PMT data requirements. Their main objective is to ensure data editors have access and are maintaining data in accordance with the organizations needs and application requirements.

Data Editors

Data Editors have the greatest knowledge of the source data being collected and are responsible for maintaining the data to the highest detail and quality available. This group has basic computer experience, understand the PMT data requirements and coordinate with the System Managers to ensure data is entered, maintained and as accurate as possible.

Data Viewers

Data Viewers vary in their computer experience and understanding of the PMT framework and its data. This group is interested in exploring and viewing the data from various aspects and perspectives. User motivations vary as well and may involve planning, decision making, data analysis, querying to answer specific questions or a combination of all.

5. System Interfaces

The PMT framework is comprised of three major components: application, API and database. Each of these three components are required and are dependent on one another. Figure 1 depicts a high level visual diagram of the relationship between them.

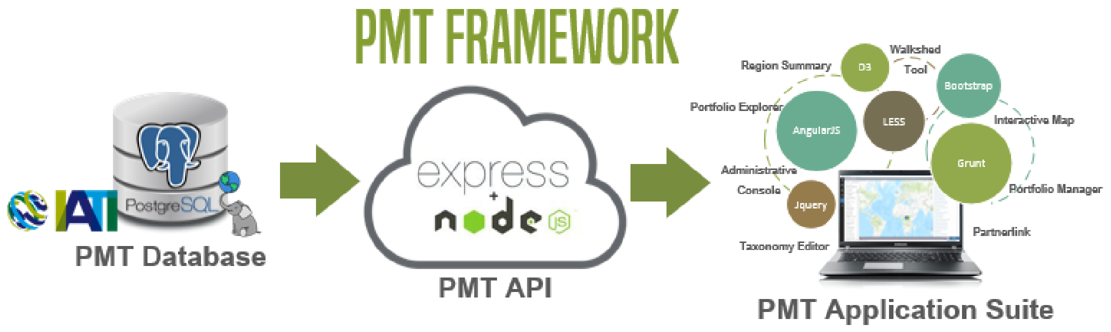

Figure 1

The PMT database is a PostgreSQL RDBMS utilizing the spatial extender PostGIS for spatial data storage support. The PMT data model is based on the IATI Standard, which provides a technical publishing framework for reporting individual development cooperation activities/projects [1].

The PMT application is a highly configurable suite of features and modules that allow users to explore, analyze, query and manage development data. The web application is built using AngularJS and common JavaScript libraries. The PMT database is accessed by the application, through a NodeJs/ExpressJs API which also provides user authentication.

6. Constraints

The PMT framework is constrained by its individual components. All three components must be able to connect to one another and be operating as intended by the developer.

The Internet connection is also a constraint, as the PMT application requires an Internet connection in order to access externally hosted basemaps and certain fonts.

The PMT application accesses all of its data through the PMT API from the PMT database, therefore is constrained by either an Intranet or Internet connection for this communication.

The PMT application has been developed and designed using HTML5 and newer JavaScript libraries. To ensure a quality user experience the browsers listed in Table 1 are suggested for user access to the application.

Table 1

Browser	Website	Supported Versions
Chrome	https://www.google.com/chrome/	48.0.2564 and higher
Mozilla Fire Fox	https://www.mozilla.org/en-US/firefox/new/?f=97	44 and higher
Internet Explorer [3]	https://www.microsoft.com/en-us/download/internet-explorer.aspx	11

7. Hardware Requirements

The hardware requirements will vary depending on whether the PMT framework is implemented on a single host or in a multiple host environment. Each component (application, API, database) can be installed on a separate host, or can be installed together on a single host. The hardware requirements for each component are included separately, to provide technical staff the information needed to procure the necessary hardware for a hosting environment in line with their internal hosting and network requirements. We recommend the database and API be hosted separately.

Database Hardware Requirements

Hardware requirements for the database will vary based on usage and performance expectations. The database is responsible for the majority of the processing and therefore will consume the majority of the resources required by the PMT framework.

A proper balance of CPU, memory and disk resources will provide the database with the requisite processing performance. The following are our minimum recommended hardware requirements for the PMT database in a production environment:

- 64-bit Dual Core CPU
- 64-bit Ubuntu 12.04.5 LTS (Precise Pangolin) Operating System [5]
- 2 GB Memory
- 50 GB disk space

The minimum of a dual core processor is highly advised as PostgreSQL is process based, however PostgreSQL and the PMT framework will operate on a single core processor.

The PMT was built on the 64-bit Ubuntu 12.04.5 operating system and has not been tested on any other operating systems. It is technically possible to run the PMT framework on another operating system, however is not advised without the necessary technical experience and in-depth understanding of all of the PMT components, as this would likely require adjustments to the core logic and/or technology dependencies.

It is possible to operate PMT and PostgreSQL with less than 2 GB of memory, but will have direct impacts to performance. We recommend increasing memory beyond 2 GB if available.

The disk space requirement is a minimum space requirement for the database, operating system and other PMT database related files and components. The disk space does not include needed space for backup and backup archives. It also does not include required space needed for hosting basemaps or contextual GIS data.

We do not have requirements for the type of disk used, however direct-attached storage is preferred over external storage due to performance.

Application Hardware Requirements

The PMT application uses only client-side technologies and therefore does not require a traditional server environment. The application can be hosted using a cloud storage service, such as Amazon s3. The application consumes approximately 15MB of space and can be hosted on any hardware with web server technology.

API Hardware Requirements

The PMT API is built on NodeJS and is implemented, for the usage of PMT, as a single core application. The API's main function is to facilitate communication between the PMT application and the PMT database, it does not do a large amount of server-side processing. The only effective way to increase performance of the API, via the hardware resources, would be through a memory increase. The following are minimum requirements for the PMT API in a production environment:

- 64-bit Single Core CPU
- 64-bit Ubuntu 12.04.5 LTS (Precise Pangolin) Operating System [5]
- 2 GB Memory
- 15 GB disk space

8. Software Requirements

The following are the software requirements for each component of the PMT framework. Each component has its own specific software requirements.

Database Software Requirements

Below are the software requirements for the PMT database:

- PostgreSQL 9.3 [6]
- PostGIS 2.1.0 [7]

Application Software Requirements

The PMT application does not have any software requirements. The application uses only client-side technologies, which are bundled together with the application code so that it is completely self-contained.

There are a number of software requirements required to build the application, which is required to manage and maintain the application. The application can be built on a personal computer, and does not require a server environment. The following are the software required to build the website:

- Web developer based IDE (Visual Studio Code, WebStorm, etc.)
- Node.js (v.6)
- NPM
- Bower

API Software Requirements

Below are the software requirements for the API:

- libcap2-bin [8]
- NPM
- Node.js (v.6)
- pm2 (v.2.2.1) [9]

9. Resource Capacity Requirements

Prior to implementation of the PMT framework within a client's environment, it is imperative for the client to be aware of the required skill sets and knowledge for proper and effective management of the PMT framework. The following is a list of skills and concepts the technical team should be aware of prior to taking ownership of the PMT framework:

General Concepts

- Geo-spatial theories and technologies
- IATI Standard
- Understanding of the PMT framework and dependencies

Application & API Management

- Web development best practices and technologies
- Architectural patterns (MVC, MVW, MVVM)
- Unit testing frameworks (Jasimin, Karma)
- Dependency management tools (Bower, NPM)
- JavaScript and commonly JS used libraries (AngularJs, Underscore, Jquery, D3)
- Server-side JavaScript (NodeJs)
- RESTful API concepts and technologies (ExpressJs)
- JavaScript based task automation (Grunt)
- Web-based spatial visualization and interactive mapping technologies (Leaflet, MapGL)
- UI component & responsive design frameworks (Angular Material, Bootstrap)
- CSS pre-processors (Less)

Database Management

- RDBMS concepts and best practices (indexes, triggers, stored procedures, data models, constraints)
- Experience maintaining complex data within database systems (Postgres)
- RDBMS management tools (PgAdmin)
- Procedural database languages (PL/pgSQL)
- Understanding of spatial data and management within an RDBMS (PostGIS)

Server Technologies

- Linux based operating systems (Ubuntu)
- Process manager (pm2)

10. PMT Framework & Instance Installation

The PMT Framework requires configuration for one or more application instances. The following instructions assume the installation of a single configured application instance. The framework is publically available in the SpatialDev GitHub PMT repo. The repo is separated into the three components:

- PMT-Database – contains the Postgres PMT database model, documentation and change management scripts.
- PMT-API – the NodeJS/ExpressJS API.
- PMT-Viewer – the AngularJS modular application.

Each application instance is assigned a unique instance name. The framework within the above repo requires configuration tailored for a specific instance. You should have received a app.config.js file in addition to a database backup file for this installation process. For the purposes of these directions the instance name is **spatialdev** and you should replace this with your provided instance name throughout the instructions where it appears.

The following are instructions for installing each component of the framework and are intended to be done in order: database, API and application.

Database Installation Instructions

The PMT database will be provided as a Postgres backup file with a specific naming convention: spatialdev10cs108ds88.tar

spatialdev10cs108ds88.tar – the instance name.

spatialdev**10**cs108ds88.tar – the database version.

spatialdev10**cs108**ds88.tar – the change script version (data model).

spatialdev10cs108**ds88**.tar – the data script version (data iteration).

1. Download/copy provided **spatialdev10cs108ds88.tar** file to the database server with PostgreSQL/PostGIS installed.
2. Connect to PosgreSQL via psql or pgAdmin.
3. Create a new database with instance and database version: spatialdev10
   ```
   CREATE DATABASE spatialdev10 OWNER postgres;
   ```
4. Restore database backup to new database (from server command line). Update paths to match local directories. The database will take 10-20 minutes to fully restore:
   ```
   /usr/lib/postgresql/9.3/bin/pg_restore -U postgres -d
   spatialdev10 -Ft /usr/local/spatialdev10cs108ds88.tar
   ```
5. Validate restore by connecting to psql or pgAdmin and viewing records from the activity table:
   ```
     SELECT * FROM activity LIMIT 100;
   ```
6. Cleanup pages and indexing after restoring. Execute the following, one at a time:
   ```
   VACUUM;
   ANALYZE;
   ```

Automated Database Backups

Database backups are automated through cron. Backup retention is as follows:

- Daily - runs at midnight, keep last 7 backups only
- Weekly - runs at midnight on Sunday, keep last month of Sundays
- Monthly - runs first Sunday of month at midnight, keep last year of monthly backups

Setup Instrustions

1. Using psql or pgAdmin execute the following on database postgres as user postgres to add new database user pmt (*only perform once per server*):
   `CREATE USER pmt WITH PASSWORD 'password'; ALTER USER pmt WITH SUPERUSER;`
2. Setup Postgres permissions (only perform once per server):
   a. Change user to pmt (user was created in above step): `su pmt`
   b. Create a pgpass file: `touch /home/pmt/.pgpass`
   c. Open .pgpass file for editing: `vi /home/pmt/.pgpass`
   d. Copy and past the following text into the file `*.*.*:pmt:password`
   e. Change permissions on pgpass file: `chmod 600 /home/pmt/.pgpass`
3. Create directory for backup files and grant permissions (only perform once per server):
   `mkdir /usr/local/pmt_bu chmod 777 /usr/local/pmt_bu`
4. Create a cron job for database backups (repeat for each database):
   a. Open cron for editng as user: `crontab -u pmt -e`
5. Add the following lines, as needed (*note database instance name*). Daily backups for dev environments and all three for production environments:
   a. Daily Backup @ Midnight (keeps last 7 days):
   ```
   0 0 * * * pg_dump -U pmt -Ft -w spatialdev10 >
   /usr/local/pmt_bu/spatialdev10_$(date +\%A).tar
   ```
   b. Weekly Backup @ Midnight on Sunday (keeps last month):
   ```
   0 0 * * 0 pg_dump -U pmt -Ft -w spatialdev10 >
   /usr/local/pmt_bu/spatialdev10_Week$(date +\%W).tar
   ```
   c. Monthly Backup @ Midnight on First Day (keeps last year):
   ```
   0 0 1 * * pg_dump -U pmt -Ft -w spatialdev10 >
   /usr/local/pmt_bu/spatialdev10_$(date +\%b\%Y).tar
   ```

Database Export/Import Feature and Automated Email Support

PMT provides the capability to export data in a csv or IATI xml format through database functions. All of the export functions use a file naming convention and create files in **/usr/local/pmt_dir** on the database server. Each file name begins with the email address of the intended recipient, followed by an underscore (_) and the database instance name. (i.e. *jane.doe@myemail.com_project_export.csv*)

Using incrond, a inotify cron daemon which monitors file system events, the **/usr/local/pmt_dir** is monitored for new files. When a new file is detected, a bash file is executed. The bash file extracts the email address from the file name, prepares the email, attaches the file, emails the file to the recipient and then deletes the file from the directory.

1. Install incron
   ```
   sudo apt-get install incron
   ```
2. Install postfix
   ```
   sudo apt-get install postfix mailutils libsasl2-2 ca-certificates
   libsasl2-modules
   ```
   a. The postfix configuration wizard will launch:
      i. Choose server: Internet Site
      ii. For FQDN: mail.example.com
3. Configure postfix to use Go Daddy Email:
   a. Open the postfix configuration file `sudo vim /etc/postfix/main.cf`
   b. Add the following to the bottom of the file:
   ```
   relayhost = [smtpout.secureserver.net]:80
   smtp_sasl_auth_enable = yes
   smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
   smtp_sasl_security_options = noanonymous
   smtp_tls_CAfile = /etc/postfix/cacert.pem
   smtp_use_tls = yes
   message_size_limit=20480000
   ```
   c. Open the postfix password file
   ```
      sudo vim /etc/postfix/sasl_passwd
   ```
   d. Add the following line replacing USERNAME@yourdomain.com with your email and PASSWORD with your email password. [smtpout.secureserver.net]:80 USERNAME@yourdomain.com:PASSWORD
   e. Fix the permissions
   ```
   sudo chmod 400 /etc/postfix/sasl_passwd
   ```
   f. Update postfix to use the sasl_passwd file
   ```
   sudo postmap /etc/postfix/sasl_passwd
   ```
   g. Validate certificates
   ```
   cat /etc/ssl/certs/Thawte_Premium_Server_CA.pem | sudo tee
   -a /etc/postfix/cacert.pem
   ```
   h. Restart postfix
   ```
   sudo /etc/init.d/postfix restart
   ```
4. Test postfix Replace your email address in the command below.
   ```
   echo "Test mail from postfix" | mail -s "Test Postfix"
   jane.doe@myemail.com
   ```
   a. If you do not receive an email look in the mail log for errors
   ```
   cat /var/log/mail.log
   ```
   b. If you see: warning: unable to look up public/pickup: No such file or directory then sendmail may still be running. You need to stop sendmail and restart postfix.
   ```
   sudo /etc/init.d/sendmail stop
   sudo /etc/init.d/postfix restart
   ```
5. Install mutt
   ```
   sudo apt-get install mutt
   ```
6. Test mutt Replace your email address in the command below.
   ```
   echo "My test email for mutt" | mutt jane.doe@myemail.com
   ```

7. Add user (pmt) to run incron and send emails
   `sudo adduser pmt`
8. Add pmt user as sudo user by adding the following entry to the visudo file
   a. Open visudo file `sudo /usr/sbin/visudo`
   b. Add the following to the file under *#user priviledge specification*
      `pmt ALL=(ALL:ALL) ALL`
9. Create log file, and grant permissions to user pmt to use it
   ```
   sudo touch /var/log/pmt_email.log
   sudo chmod 777 /var/log/pmt_email.log
   ```
10. Copy the *PMT/Database/Documentation/Installation/ServerProcesses/pmt_email.sh* bash file from the github repo to /usr/local/bin/pmt_email.sh and grant permissions to user pmt to use it: `sudo chmod 777 /usr/local/bin/pmt_email.sh`
11. Create directory for files and grant permissions to user pmt to use it:
    ```
    sudo mkdir /usr/local/pmt_dir
    sudo chmod 777 /usr/local/pmt_dir
    ```
12. Give permission to pmt to run incron by adding pmt to the following file
    a. Open file
       `sudo vi /etc/incron.allow`
    b. Add user pmt to the file and save
       `pmt`
13. Add the incron job:
    a. Open incron table for editing
       `incrontab -e`
    b. Add the following line and save
       ```
       /usr/local/pmt_dir IN_CREATE /usr/local/bin/pmt_email.sh $@
       $# $%
       ```
14. Start incron service `sudo service incron start`
15. Test process by adding a file to the watched directory. Replace your email address in the command below.
    `sudo touch /usr/local/pmt_dir/jane.doe@mymail.com_pmt.txt`

API Installation Instructions

The PMT API is a NodeJS/ExpressJS application that can be installed with npm directly from the repo and run on the server using pm2.

API Download and Build

1. To install Node.js, type the following command in your terminal:
   `sudo apt-get install nodejs`
2. Then install the Node package manager, npm:
   `sudo apt-get install npm`
3. Create a symbolic link for node, as many Node.js tools use this name to execute:
   `sudo ln -s /usr/bin/nodejs /usr/bin/node`
4. Now we should have both the Node and npm commands working:
   ```
   $ node -v
   $ npm -v
   ```

5. Clone the PMT-API repo to the server:
```
git clone git://github.com/spatialdev/PMT
cd PMT/API
```
6. Install the required packages:
```
$ npm install
```
7. Open the /PMT/API/config.js in an editor and update the **bold** items appropriately:
```
 // Postgres settings
 pg: {
     // Production Server: spatialdev10
     1: {
         host: "123.45.678.9",
         port: "5432",
         user: "postgres",
         password: "postgrespassword",
         database: "spatialdev10",
         smpt: {
             host: 'smtp.gmail.com',
             port: 465,
             secure: true, // use SSL
             auth: {
                 user: 'yourcompnay@gmail.com',
                 pass: 'gmailpassword'
             }
         }
     }
   }
```
8. To launch the API using your local environment:
```
node api.js local
```
9. Ctrl+c to stop after confirming execution without error.

PM2 Installation and API Setup

1. First, we do this so we do not have to be sudo to run an app on port 80.
```
sudo apt-get install libcap2-bin
sudo setcap cap_net_bind_service=+ep /usr/bin/nodejs
```
2. Currently, our pm2 setup is very simple. Set port to 80 in config.js. Then, we install pm2 this way:
```
sudo npm install -g pm2
```
3. We want pm2 to be a daemon and restart our app if it crashes or if the server restarts:
```
sudo su -c "env PATH=$PATH:/usr/bin pm2 startup linux -u ubuntu"
```
4. Then vim into `/etc/init.d/pm2-init.sh` and change `PM2_HOME` to `/home/ubuntu/.pm2`.
5. Then we want to start the app in pm2. Create a json file named `ecosystem.config.js` and copy in the below. **Make sure the path to the script is correct, relevant to this file**:
```
{
  "apps" : [
  {
    "name": "pmt-api",
    "script"     : "PMT/API/api.js",
    "instances" : 3,
    "exec_mode" : "cluster"
        "args"     : ["production"],
    "node_args" : ["production"],
    "env"        : {
       "NODE_ENV":"production"
    }
   }
  ]
}
```

6. Make pm2 reload this file:
   `pm2 reload ecosystem.config.js --update-env`
7. Show the running applications:
   `pm2 list`

Useful pm2 Commands

- To stop all process:
  `pm2 stop all`
- To restart one application "pmt-api":
  `pm2 restart pmt-api`
- To delete one application "pmt-api":
  `pm2 delete pmt-api`
- View logs and application details for "pmt-api":
  `pm2 show pmt-api`
- If you want to update to the latest commit and redeploy:
  ```
  cd /home/ubuntu/PMT/API/
  git pull origin master
  pm2 restart pmt-api
  ```

Application Installation Instructions

The PMT Viewer is an AngularJS NodeJS application that can be installed with npm directly from the repo and built using GruntJS.

Download and Install the Application

1. Change directories to the PMT Viewer:
   `cd PMT/Viewer/Application`
2. Install the required packages and Grunt:
   ```
   $ sudo npm -g install grunt-cli karma bower
   $ sudo npm install
   $ bower install
   ```
3. Replace the app.config.js file in PMT/Viewer/Application/ with your provided file. Then open the file and update the following json value for the parameter **spatialdev.constants.pmt.api.production** with either your **domain** (where you plan to host the application) or with the **server's IP** address:
   `"production": "http://`**`api.v10.investmentmapping.org`**`:8080/api/"`
4. Build the application for your instance:

```
grunt build -env=production -theme=spatialdev
```

5. Copy the contents of the PMT/Viewer/Application/build and publish it to your web server of your choice (*index.html as target*).

References

[1] "The IATI Standard". *International Aid Transparency Initiative (IATI)*. N.p., 2017. Web. 18 Jan. 2017.

[2] Gislounge. "Basemaps Defined ~ GIS Lounge." *GIS Lounge*. N.p., 11 July 2013. Web. 18 Jan. 2017.

[3] "Support for older versions of Internet Explorer ended." *Internet Explorer End of Support.* N.p., n.d. Web. 18 Jan. 2017.

[4] "Host (network)." *Wikipedia.* Wikimedia Foundation, n.d. Web. 18 Jan. 2017.

[5] "Ubuntu 12.04.5 LTS (Precise Pangolin)." *Ubuntu 12.04.5 LTS (Precise Pangolin).* N.p., n.d. Web. 18 Jan. 2017.

[6] "E.16. Release 9.3." *PostgreSQL: Documentation: 9.3: Release 9.3*. N.p., n.d. Web. 18 Jan. 2017.

[7] Developers, PostGIS. "PostGIS Project Steering Committee (PSC)." — *PostGIS 2.1.0 Released*. N.p., n.d. Web. 18 Jan. 2017.

[8] "Package: libcap2-bin (1:2.22-1ubuntu3)." *Ubuntu – Details of package libcap2-bin in precise.* N.p., n.d. Web. 18 Jan. 2017.

[9] "PM2 ·." *PM2 - Advanced Node.js process manager*. N.p., n.d. Web. 18 Jan. 2017.