



# Internet Voting System

Architectural Description



Vilnius, Lithuania  
Version: 1.0

December 10th, 2018  
**Kęstutis Matuliauskas**

info@digitaldemocracy.io  
DigitalDemocracy.io

# Table of Contents

<b>1.....Executive Summary</b>	<b>3</b>
1.1 Introduction	
1.2 Technologies	
1.3 Commercial release goals:	
<b>2.....General architectural principles</b>	<b>4</b>
2.1 A distributed system	
2.2 Cloud / dedicated server use	
2.3 Multitenancy (also known as “Partners support”)	
<b>3.....Stakeholders</b>	<b>5</b>
3.1 List of all Stakeholders	
3.2 List of Proxy-Stakeholders	
<b>4.....Use Cases</b>	<b>7</b>
<b>5Cross-Cutting Architectural design decisions</b>	<b>7</b>
5.1 WordPress CMS – Cross-Cutting Architectural Decision	
<b>6.....Viewpoints &amp; Principles</b>	<b>8</b>
6.1 Viewpoints used	
6.2 Principles used	
<b>7.....Context view</b>	<b>9</b>
7.1 Main Diagram	
7.2 Involved stakeholders	
<b>8.....Functional View</b>	<b>10</b>
8.1 Main functions diagram	
8.2 Security-critical functions (Security perspective)	
<b>9.....Information View</b>	<b>11</b>
9.1 SQL Schema of main tables	
9.2 SQL Tables relationships	
9.3 Involved Stakeholders	
9.4 BCNF database structure - Architectural Decision for Information View	
9.5 Information ownership grid (Security perspective)	
9.6 Data Storage (Availability and Resilience perspective)	
9.7 From Legal Perspective	
9.8 From Evolution Perspective	
9.9 From Performance & Scalability Perspective	
9.10 From Security Perspective	
9.11 From Availability & Resilience (Recovery) Perspective	

<b>10.....</b>	<b>Concurrency View</b>	<b>20</b>
10.1	Process model (Performance & Scalability perspective)	
10.2	Principles used from Performance & Scalability perspective	
<b>11.....</b>	<b>Development View</b>	<b>21</b>
11.1	System models and main interfaces	
11.2	Class diagrams	
11.3	S.O.L.I.D. MVC - Architectural Decision for Development View & Information Views	
11.4	SemVer - Architectural Decision for Development & Information Views	
<b>12.....</b>	<b>Deployment View</b>	<b>27</b>
12.1	Network & servers infrastructure	
12.2	Principles used from Security perspective	
12.3	Principles used from Legal perspective	
<b>13.....</b>	<b>Operational View</b>	<b>29</b>
13.1	Stakeholders	
13.2	Principles used from Performance & Scalability perspective	
13.3	Principles used from Availability & Resilience (Recovery) perspective	
13.4	Principles used from Security Perspective	
13.5	Principles used from Legal Perspective	

# 1 Executive Summary

---

## 1.1 Introduction

This is an internet voting system (a WordPress plugin) made to process local-elections primary via internet platform (website).

Internet Voting System is based on “PHP OOP MVC + Templating” architectural implementation, created by Kestutis Matuliauskas and known as “S.O.L.I.D. MVC”, which is based on know-how MVC leaders in PHP written articles regarding possible MVC implementation in PHP. That architectural decision was used for other project, and it is appears to be a stable and valid enable.

“S.O.L.I.D. MVC” is be released under MIT license, that, differently to GPLv2/GPLv3 or AGPL, it fits for both types of projects – non-commercial and commercial use. This means that technology users can create both - commercial & non-commercial products on given architectural pattern code without the need to release the created code.

## 1.2 Technologies

### 1.2.1 Stack / IDE

Project will be create use PHP programming language (version 7.1), MySQL / MariaDB Database, WordPress CMS, that work well as REST API and Development & Testing framework as well.

For testing will be used PHPUnit 6.4, Windows 10, and PHP Storm IDE with PHP7 Zend Engine CodeCoverage support.

### 1.2.2 Debugging

For debugging will be used XAMPP for Win 10 with PHP 7.1, and XDebug 2.5.4 (that supports PHP 7.1). XDebug will be called via PHP Storm IDE on Windows 10 without any virtual machine or shell implementation.

On the front-end testing it will required to learn QUnit testing (creators of jQuery + official tool used in WordPress), Silex.JS for QUnit (for fake server support in Ajax queries, fake timer, and javascript alert content reading inline via JS without popups).

This is needed because “SampleMVC” will need to have his quick-starter “SampleMVC\_Tests” plugin as well, that can work immediately.

### 1.2.3 UI

For visuals SCSS won't be used. Instead of that CSS4-Variables will be used that is working much easier, supported by all major browsers (86+% market share). Problematic is only IE11 (3,5% market share). Hopefully either MS will release the IE11 update for CSS4-Variables, as it is still officially supported browser, or it's use will drop below 1% in the world by April 1, 2019, or IE11 users will still see one theme (theming for them won't work).

## 1.3 Commercial release goals:

Based on SampleMVC with Voter / Vote / Poll example and dual screen & E-mail or Twilio.com SMS confirmation, implement higher quality product, price it for 55 USD / sale. Commercial product, at least must have a great design, and its own settings dashboard, as well as logs system.

The system must have separation of Vote from Voter via separated DB tables.

## 2 General architectural principles

---

### 2.1 A distributed system

For security (OS-specific vulnerabilities) & stability (DDoS attacks) the system has always to have two instances running on at least two different platforms (Windows & UNIX) at least two different OSes (Ubuntu Server & Windows Server), on at least two different physical TIER1 datacenter locations in EEA (Europe Economic Area: European Union + Switzerland & UK), i.e. Microsoft Datacenter TIER1+ in Dublin, Ireland with database-hosting in Lithuania for Windows Azure and “Cherry Servers” TIER1 data center Siauliai, Lithuania.

### 2.2 Cloud / dedicated server use

The plugin together with the system will stay in at least Microsoft Azure cloud with support of PHP7 and MySQL/MariaDB, probably on one of Windows Server distributions and the Siauliai Data Center on Ubuntu Server and Plesk server management tool that is best-suited for WordPress.

### 2.3 Multitenancy (also known as “Partners support”)

If allowed by WordPress engine and plugin settings, the plugin will allow front-end registrations and allow to create voting for other people and organizations besides the system owner.

## 3 Stakeholders

### 3.1 List of all Stakeholders

Stakeholder Class	Concerns of Stakeholder
<p><b>Product lead, owner &amp; lead maintainer</b>  <b>Role name:</b> CEO (Chief Executive Officer) &amp; Software Architect</p> <p><b>Notes:</b>  <i>This is me, the company's CEO &amp; Software Architect</i></p>	<ol style="list-style-type: none"> <li>1. Business Strategy</li> <li>2. Company policy (partners to work with)</li> <li>3. Coding standards</li> <li>4. Software architecture</li> <li>5. 3<sup>rd</sup> party services to integrate</li> </ol>
<p><b>End-Users</b></p> <p><b>Notes:</b>  <i>These are the people who use the product, which is installed on buyer's company servers</i></p>	<ol style="list-style-type: none"> <li>1. Vote in elections via product</li> <li>2. Give a feedback to buyer's company about issues with the system, so that they could report that to product creator's (seller's) company.</li> </ol>
<p><b>All developers in the company</b>  <b>Role names:</b> Sub-System's Lead Developers, Senior Developers, Summer Interns (Junior Developers)</p> <p><b>Notes:</b>  <i>There are the maintainers as well.</i></p>	<ol style="list-style-type: none"> <li>1. Write the code.</li> <li>2. Learn coding standards.</li> </ol>
<p><b>#1 (main) paying customer (organization)</b>            Entity example: "California State Government", "Vilnius City Municipality"</p> <p><b>Notes:</b>  <i>By the majority of the needs of this customer the system primary architecture is based.</i></p>	<ol style="list-style-type: none"> <li>1. Be the main tester of the system.</li> <li>2. Be the main customer of the system.</li> <li>3. Run the system for elections.</li> </ol>
<p><b>All rest paying customers</b></p> <p><b>Notes:</b>  <i>These are the buyers of the dual licensed commercial plugin via Envato marketplace with ThemeForest Split license with Envato and AGPL license). We pay attentions to their needs and prioritize them.</i></p>	<ol style="list-style-type: none"> <li>1. Use the system.</li> <li>2. Report bugs.</li> </ol>
<p><b>Non-paying users of free version</b></p> <p><b>Notes:</b>  <i>These are the downloaders of minimalistic version "Guest Polls" W.org free AGPL plugin.</i></p>	<ol style="list-style-type: none"> <li>1. Try the system framework, share experience, tell what are the missing parts to buy the commercial software is.</li> </ol>

Stakeholder Class	Concerns of Stakeholder
<b>Main marketing person</b> <b>Role name:</b> CMO (Chief Marketing Officer)  <b>Notes:</b> <i>This is the Google AdWords-certifier guy who spends fixed company budget buying links on Google main search page as well as he does the remarketing with plugin's banners across the internet.</i>	<ol style="list-style-type: none"> <li>1. Buy ads on Google for Google AdWords.</li> </ol>
<b>Network (Server) administrators</b> <b>Notes:</b> <i>Mostly expected that those are hired by buyer's company.</i>	<ol style="list-style-type: none"> <li>1. Make sure all servers are up-to-date firmware-wise, OS-wise, WordPress-wise, WP plugins-wise</li> </ol>
<b>Testers</b>	<ol style="list-style-type: none"> <li>1. Make sure all major functions are working.</li> </ol>
<b>Website administrators</b> <b>Notes:</b> <i>Mostly expected that those are hired by buyer's company.</i>	<ol style="list-style-type: none"> <li>1. Insert decision makers.</li> <li>2. Add voting.</li> <li>3. Void false votes.</li> <li>4. Investigate potential risks and report to server administrations.</li> </ol>

### 3.2 List of Proxy-Stakeholders

Stakeholder	Concerns of Stakeholder
<b>Main copywriter &amp; copywriters manager</b> <b>Role name:</b> CCO (Chief Content Officer)  <b>Notes:</b> <i>This is the guy who writes all the SEO articles on company's website about how great the system is, as well as he communicates the Forbes.com, NYTimes.com and "HuffPost" (HuffingtonPost.com) writers that writes the articles how the great &amp; secure the plugin / system is.</i>	<ol style="list-style-type: none"> <li>1. Write articles on Forbes.com, NYTimes.com, HuffingtonPost.com and other websites.</li> </ol>
<b>Partners for premium support &amp; reselling</b> Example: GreatSupportCompany.com  <b>Notes:</b> <i>These are the companies (preferred) / individuals (possible scenario) who do individual/premium support (skype calls, 24/7 support etc.) for the end-customers as well as they are one who make the website/network-setup for end-customers, that includes the buying of "Internet Voting System" license for ever customer.</i>	<ol style="list-style-type: none"> <li>1. Boost the sales.</li> <li>2. Do a premium support.</li> </ol>

## 4 Use Cases

---

Main system use cases

1. Vote securely in elections.
2. Via fork of petitions extension, it can be used for petitions as well.

## 5 Cross-Cutting Architectural design decisions

---

### 5.1 WordPress CMS – Cross-Cutting Architectural Decision

1. A content management system or programming language framework to run the Internet Voting System
  - 1.1. Status: accepted
  - 1.2. Deciders: CEO, Software Architect
  - 1.3. Date: Dec 10, 2018
2. Context and Problem Statement
3. Decision Drivers
  - 3.1. A need for easy-to-start system
  - 3.2. A need for highly maintainable system
  - 3.3. A system that works for many server platforms, providers
4. Considered Options
  - 4.1. WordPress
  - 4.2. Symfony
  - 4.3. System on .NET
5. Decision Outcome
  - 5.1. Chosen option 1 – WordPress, because that is the only easy-to-start, highly maintainable system.
6. Pros and Cons of the Options
  - 6.1. Pros – Will fit and do the job as expected
  - 6.2. Cons – It requires often update



## 6 Viewpoints & Principles

---

### 6.1 Viewpoints used

Informational (SQL Schema), Functional, Informational, Deployment (Infrastructure), Operational (Live System & Premium Support Partners Assistance) viewpoints used.

### 6.2 Principles used

Legal (Regulation), Performance & Scalability, Security, Availability & Resilience (Recovery), Evolution (Change) perspectives used.

# 7 Context view

## 7.1 Main Diagram

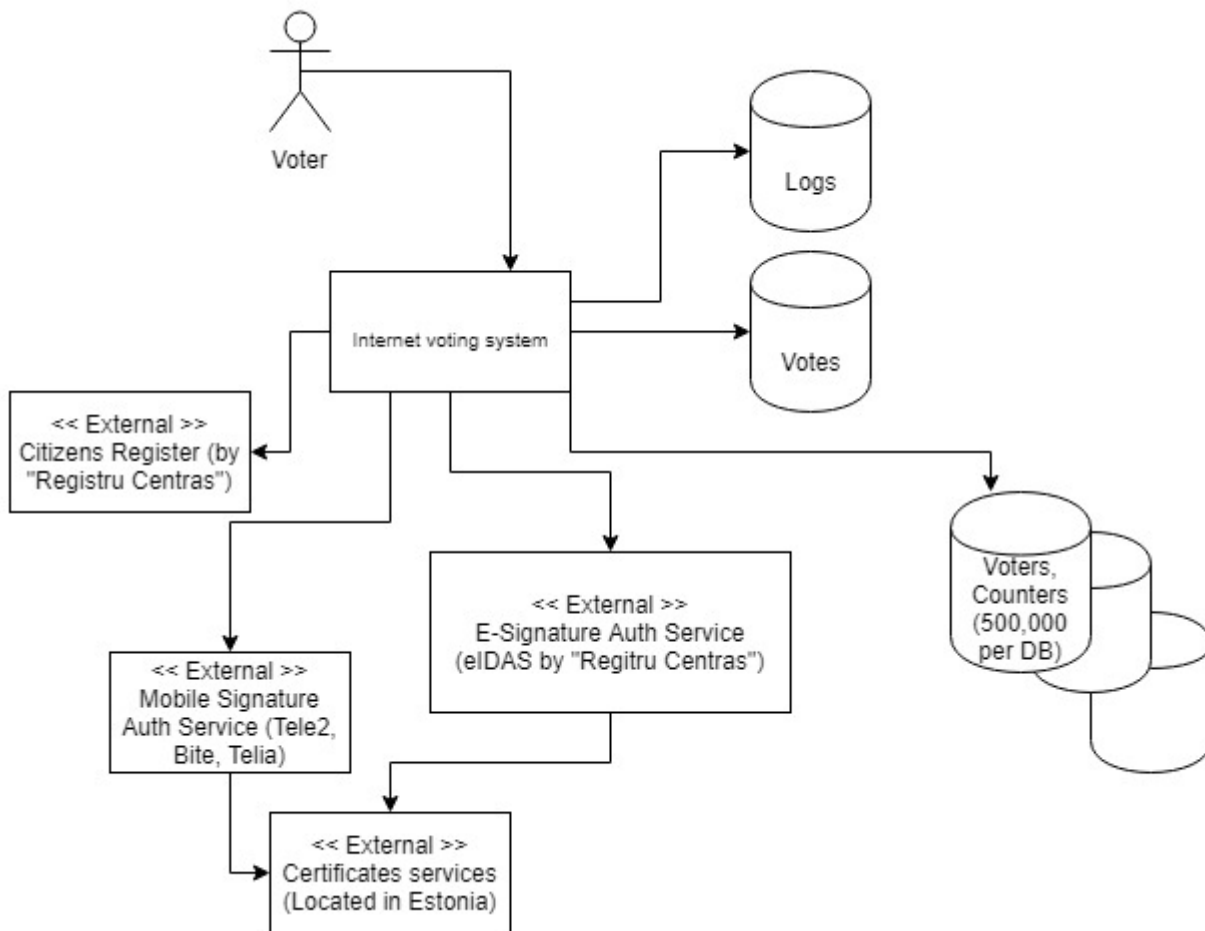


Image 1. System Context Diagram

## 7.2 Involved stakeholders

All stakeholders

## 8 Functional View

---

### 8.1 Main functions diagram

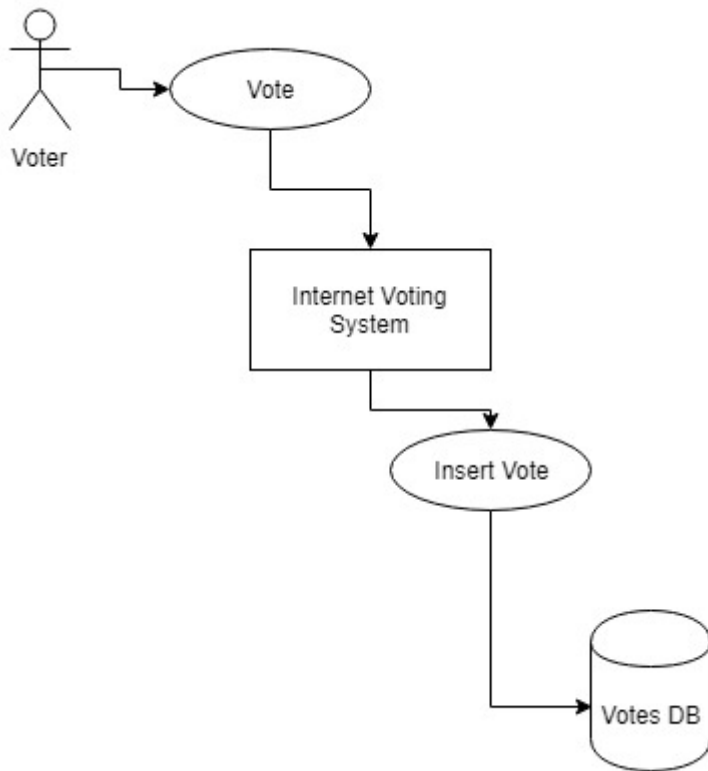


Image 2. Main system functions

### 8.2 Security-critical functions (Security perspective)

- Authorizations – managed by WordPress
- Authentication – managed by WordPress
- Permissions – set, validated & changed by the Internet Voting System, processed via WordPress CMS.
- Smooth & fast upgrade – Internet Voting System plugin supports smooth and fast upgrading.

## 9 Information View

### 9.1 SQL Schema of main tables

TABLE: Decision Makers

```
`decision_maker_id` int(11) NOT NULL AUTO_INCREMENT,  
`decision_maker_unique_identifier` varchar(20) NOT NULL,  
`partner_id` int(11) unsigned NOT NULL DEFAULT '0',  
`decision_maker_page_id` int(11) unsigned NOT NULL DEFAULT '0',  
`decision_maker_name` varchar(100) NOT NULL,  
`decision_maker_image` varchar(255) NOT NULL,  
`demo_decision_maker_image` tinyint(1) unsigned NOT NULL DEFAULT '0',  
`street_address` varchar(50) NOT NULL,  
`city_id` int(11) unsigned NOT NULL DEFAULT '0',  
`state_id` int(11) unsigned NOT NULL DEFAULT '0',  
`zip_code` varchar(20) NOT NULL,  
`country_code` varchar(2) NOT NULL,  
`phone` varchar(50) NOT NULL,  
`phone_hours` varchar(50) NOT NULL,  
`email` varchar(50) NOT NULL,  
`facebook` varchar(100) NOT NULL,  
`twitter` varchar(100) NOT NULL,  
`linkedin` varchar(100) NOT NULL,  
`display_in_dropdown` tinyint(1) unsigned NOT NULL DEFAULT '1' COMMENT 'Only  
manager can set decision maker to be displayed between dropdown options',  
`display_in_list` tinyint(1) unsigned NOT NULL DEFAULT '1' COMMENT 'Only manager  
can set decision maker to be displayed in the list',  
`decision_maker_order` int(11) unsigned NOT NULL DEFAULT '1' COMMENT 'Only manager  
can set location order',  
`ext_code` varchar(20) NOT NULL,  
`blog_id` int(11) unsigned NOT NULL DEFAULT '0',  
PRIMARY KEY (`decision_maker_id`),  
KEY `decision_maker_unique_identifier` (`decision_maker_unique_identifier`),  
KEY `partner_id` (`partner_id`),  
KEY `decision_maker_page_id` (`decision_maker_page_id`),  
KEY `city_id` (`city_id`),  
KEY `state_id` (`state_id`),  
KEY `zip_code` (`zip_code`),  
KEY `country_code` (`country_code`),  
KEY `display_in_dropdown` (`display_in_dropdown`),  
KEY `display_in_list` (`display_in_list`),  
KEY `decision_maker_order` (`decision_maker_order`),  
KEY `ext_code` (`ext_code`),  
KEY `blog_id` (`blog_id`)
```

TABLE: Entries (Entry = Voting)

```

`entry_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`entry_unique_identifier` varchar(20) NOT NULL,
`partner_id` int(11) unsigned NOT NULL DEFAULT '0',
`entry_page_id` int(11) unsigned NOT NULL DEFAULT '0',
`category_id` int(11) unsigned NOT NULL DEFAULT '0',
`entry_name` varchar(100) NOT NULL,
`entry_image` varchar(255) NOT NULL,
`demo_entry_image` tinyint(1) unsigned NOT NULL DEFAULT '0',
`entry_description` text NOT NULL,
`min_allowed_age` tinyint(2) unsigned NOT NULL DEFAULT '18',
`max_allowed_age` tinyint(2) NOT NULL DEFAULT '-1' COMMENT 'Supports [-1]
(Unlimited)',
`start_timestamp` int(11) unsigned NOT NULL DEFAULT '0',
`end_timestamp` int(11) unsigned NOT NULL DEFAULT '0',
`max_selected_options_per_response` int(11) unsigned NOT NULL DEFAULT '1' COMMENT
'No index needed for this column as we never use it in search',
`responses_goal` int(11) NOT NULL DEFAULT '1' COMMENT 'Supports [-1] (Infinite)',
`agreement_button_label` varchar(50) NOT NULL DEFAULT '',
`disagreement_button_label` varchar(50) NOT NULL DEFAULT '',
`neutrality_button_label` varchar(50) NOT NULL DEFAULT '',
`display_in_slider` tinyint(1) unsigned NOT NULL DEFAULT '1',
`display_in_list` tinyint(1) unsigned NOT NULL DEFAULT '1',
`date_created` int(11) unsigned NOT NULL DEFAULT '0',
`ext_code` varchar(20) NOT NULL,
`blog_id` int(11) unsigned NOT NULL DEFAULT '0',
PRIMARY KEY (`entry_id`),
KEY `entry_unique_identifier` (`entry_unique_identifier`),
KEY `partner_id` (`partner_id`),
KEY `entry_page_id` (`entry_page_id`),
KEY `category_id` (`category_id`),
KEY `entry_name` (`entry_name`),
KEY `min_allowed_age` (`min_allowed_age`),
KEY `max_allowed_age` (`max_allowed_age`),
KEY `period` (`start_timestamp`,`end_timestamp`),
KEY `responses_goal` (`responses_goal`),
KEY `display_in_slider` (`display_in_slider`),
KEY `display_in_list` (`display_in_list`),
KEY `date_created` (`date_created`),
KEY `ext_code` (`ext_code`),
KEY `blog_id` (`blog_id`)

```

TABLE: Entry Decision Makers (Entry = Voting)

```
`entry_id` int(11) unsigned NOT NULL DEFAULT '0',  
`decision_maker_id` int(11) unsigned NOT NULL DEFAULT '0',  
`ext_code` varchar(20) NOT NULL,  
`blog_id` int(11) unsigned NOT NULL DEFAULT '0',  
KEY `entry_id` (`entry_id`),  
KEY `decision_maker_id` (`decision_maker_id`),  
KEY `ext_code` (`ext_code`),  
KEY `blog_id` (`blog_id`)
```

TABLE: Entry Options (Entry = Voting)

```
`option_id` int(11) unsigned NOT NULL AUTO_INCREMENT,  
`entry_id` int(11) unsigned NOT NULL DEFAULT '0',  
`option_name` varchar(100) NOT NULL,  
`ext_code` varchar(20) NOT NULL,  
`blog_id` int(11) unsigned NOT NULL DEFAULT '0',  
PRIMARY KEY (`option_id`),  
KEY `entry_id` (`entry_id`),  
KEY `ext_code` (`ext_code`),  
KEY `blog_id` (`blog_id`)
```

TABLE: Respondents

```

`respondent_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`account_id` int(11) unsigned NOT NULL DEFAULT '0' COMMENT 'For WordPress it is
WP_USER_ID',
`special_status_code` varchar(20) NOT NULL,
`title` varchar(5) NOT NULL,
`first_name` varchar(50) NOT NULL,
`last_name` varchar(50) NOT NULL,
`birthdate` date NOT NULL DEFAULT '0000-00-00',
`street_address` varchar(50) NOT NULL,
`city` varchar(50) NOT NULL,
`state` varchar(50) NOT NULL,
`zip_code` varchar(20) NOT NULL,
`country_code` varchar(2) NOT NULL,
`phone` varchar(50) NOT NULL,
`email` varchar(50) NOT NULL,
`additional_1` varchar(255) NOT NULL,
`additional_2` varchar(255) NOT NULL,
`additional_3` varchar(255) NOT NULL,
`additional_4` varchar(255) NOT NULL,
`additional_5` varchar(255) NOT NULL,
`total_responses` int(11) unsigned NOT NULL DEFAULT '0',
`confirmed_responses` int(11) unsigned NOT NULL DEFAULT '0',
`voided_responses` int(11) unsigned NOT NULL DEFAULT '0',
`date_created_timestamp` int(11) unsigned NOT NULL DEFAULT '0',
`last_used_timestamp` int(11) unsigned NOT NULL DEFAULT '0',
`date_created_ip` varchar(45) NOT NULL DEFAULT '0.0.0.0',
`date_created_real_ip` varchar(45) NOT NULL DEFAULT '0.0.0.0',
`last_used_ip` varchar(45) NOT NULL DEFAULT '0.0.0.0',
`last_used_real_ip` varchar(45) NOT NULL DEFAULT '0.0.0.0',
`host` varchar(255) NOT NULL,
`agent` varchar(255) NOT NULL,
`browser` varchar(50) NOT NULL,
`os` varchar(50) NOT NULL,
`resolution` varchar(11) NOT NULL DEFAULT '0x0' COMMENT 'Maximum supported is
99999x99999',
`locked` tinyint(1) unsigned NOT NULL DEFAULT '0',
`blog_id` int(11) unsigned NOT NULL DEFAULT '0',
PRIMARY KEY (`respondent_id`),
KEY `account_id` (`account_id`),
KEY `special_status_code` (`special_status_code`),
KEY `first_name` (`first_name`),
KEY `last_name` (`last_name`),
KEY `identifier` (`birthdate`,`email`) COMMENT 'One e-mail can have multiple
family members with different birthdates',
KEY `total_responses` (`total_responses`),
KEY `date_created_timestamp` (`date_created_timestamp`),
KEY `last_used_timestamp` (`last_used_timestamp`),
KEY `last_used_ip` (`last_used_ip`),
KEY `last_used_real_ip` (`last_used_real_ip`),
KEY `date_created_ip` (`date_created_ip`),
KEY `date_created_real_ip` (`date_created_real_ip`),
KEY `locked` (`locked`),
KEY `blog_id` (`blog_id`)

```

Table: Public Responses (Response = Vote)

```

`response_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`response_hash` varchar(50) DEFAULT NULL,
`response_timestamp` int(11) unsigned NOT NULL DEFAULT '0',
`confirmation_timestamp` int(11) unsigned NOT NULL DEFAULT '0',
`voidance_timestamp` int(11) unsigned NOT NULL DEFAULT '0',
`expiration_timestamp` int(11) NOT NULL DEFAULT '-1' COMMENT 'Supports [-1]
(Never expires)',
`decision_maker_unique_identifier` varchar(20) NOT NULL,
`area_code` varchar(20) NOT NULL,
`city_code` varchar(20) NOT NULL,
`state_code` varchar(20) NOT NULL,
`zip_code` varchar(20) NOT NULL,
`country_code` varchar(20) NOT NULL,
`age_group_id` int(11) NOT NULL DEFAULT '-1' COMMENT 'Special one-to-one
relationship parameter that may impact total price. Supports [-1] (Ignore).',
`special_status_id` int(11) NOT NULL DEFAULT '-1' COMMENT 'Special one-to-one
relationship parameter that may impact total price. Supports [-1] (Ignore).',
`respondent_id` int(11) unsigned NOT NULL DEFAULT '0',
`status` varchar(20) NOT NULL DEFAULT 'UNCONFIRMED' COMMENT 'UNCONFIRMED,
CONFIRMED or VOIDED',
`response_ip` varchar(45) NOT NULL DEFAULT '0.0.0.0',
`response_real_ip` varchar(45) NOT NULL DEFAULT '0.0.0.0',
`ext_code` varchar(20) NOT NULL,
`blog_id` int(11) unsigned NOT NULL DEFAULT '0',
PRIMARY KEY (`response_id`),
KEY `response_hash` (`response_hash`),
KEY `respondent_id` (`respondent_id`),
KEY `expiration_timestamp` (`expiration_timestamp`),
KEY `response_timestamp` (`response_timestamp`),
KEY `decision_maker_unique_identifier` (`decision_maker_unique_identifier`),
KEY `area_code` (`area_code`),
KEY `city_code` (`city_code`),
KEY `state_code` (`state_code`),
KEY `zip_code` (`zip_code`),
KEY `country_code` (`country_code`),
KEY `status` (`status`),
KEY `response_ip` (`response_ip`),
KEY `response_real_ip` (`response_real_ip`),
KEY `ext_code` (`ext_code`),
KEY `blog_id` (`blog_id`)

```



Table: Secret Response (Response = Vote)

```
`response_id` int(11) unsigned NOT NULL AUTO_INCREMENT,  
`response_hash` varchar(50) DEFAULT NULL,  
`response_timestamp` int(11) unsigned NOT NULL DEFAULT '0',  
`confirmation_timestamp` int(11) unsigned NOT NULL DEFAULT '0',  
`voidance_timestamp` int(11) unsigned NOT NULL DEFAULT '0',  
`expiration_timestamp` int(11) NOT NULL DEFAULT '-1' COMMENT 'Supports [-1]  
(Never expires)',  
`decision_maker_unique_identifier` varchar(20) NOT NULL,  
`area_code` varchar(20) NOT NULL,  
`city_code` varchar(20) NOT NULL,  
`state_code` varchar(20) NOT NULL,  
`zip_code` varchar(20) NOT NULL,  
`country_code` varchar(20) NOT NULL,  
`status` varchar(20) NOT NULL DEFAULT 'UNCONFIRMED' COMMENT 'UNCONFIRMED,  
CONFIRMED or VOIDED',  
`ext_code` varchar(20) NOT NULL,  
`blog_id` int(11) unsigned NOT NULL DEFAULT '0',  
PRIMARY KEY (`response_id`),  
KEY `response_hash` (`response_hash`),  
KEY `expiration_timestamp` (`expiration_timestamp`),  
KEY `response_timestamp` (`response_timestamp`),  
KEY `decision_maker_unique_identifier` (`decision_maker_unique_identifier`),  
KEY `area_code` (`area_code`),  
KEY `city_code` (`city_code`),  
KEY `state_code` (`state_code`),  
KEY `zip_code` (`zip_code`),  
KEY `country_code` (`country_code`),  
KEY `status` (`status`),  
KEY `ext_code` (`ext_code`),  
KEY `blog_id` (`blog_id`)
```

## 9.2 SQL Tables relationships

1 x Respondent → 1 x (Entry [TYPE=PUBLIC], Public Response)

1 x Respondent → 1 x (Entry [TYPE=SECRET], Secret Response)

1 x Entry → N x Decision Makers

1 x Decision Maker → N x Entries

1 x Entry [TYPE=PUBLIC] → N x Public Responses

1 x Entry [TYPE=SECRET] → N x Secret Responses

\*Respondent = Voter; Entry = Voting; Response = Vote;

## 9.3 Involved Stakeholders

- ✓ CEO
- ✓ Software Architect
- ✓ Senior Developers
- ✓ Testers

## 9.4 BCNF database structure - Architectural Decision for Information View

1. BCNF database structure
  - 1.1. Status: accepted
  - 1.2. Deciders: CEO, Software Architect
  - 1.3. Date: Dec 10, 2018
2. Context and Problem Statement
3. Decision Drivers
  - 3.1. Easy to use, easy to scale, easy to read
4. Considered Options
  - 4.1. BCNF
  - 4.2. Non-BCNF database structure
  - 4.3. NoSQL database
5. Decision Outcome
 

Chosen option 1

  - 5.1. Pros: Scales well, easy to read
  - 5.2. Cons: May take more space
6. Pros and Cons of the Options
  - 6.1. Pros: Scales well, easy to read
  - 6.2. Cons: May take more space

## 9.5 Information ownership grid (Security perspective)

Table \ Stakeholder	Respondent (i.e. Voter)	Website Administrator
<b>Decision Makers</b> <i>I.e. San Francisco State Municipality</i>	View	View / Add / Edit
<b>Entries</b> <i>I.e. City community elder election</i>	View	View / Add / Edit
<b>Respondents</b> <i>I.e. Voters</i>	Self-view	View / Add / Edit
<b>Public Responses</b> <i>i.e. Voter' votes, if voting is not secret</i>	One-way insert by response	View / Can set as Voided
<b>Secret Responses</b> <i>i.e. Voter' votes on secret voting</i>	One-way insert by response	View / Can set as Voided
<b>Settings</b>	Read	Edit

## 9.6 Data Storage (Availability and Resilience perspective)

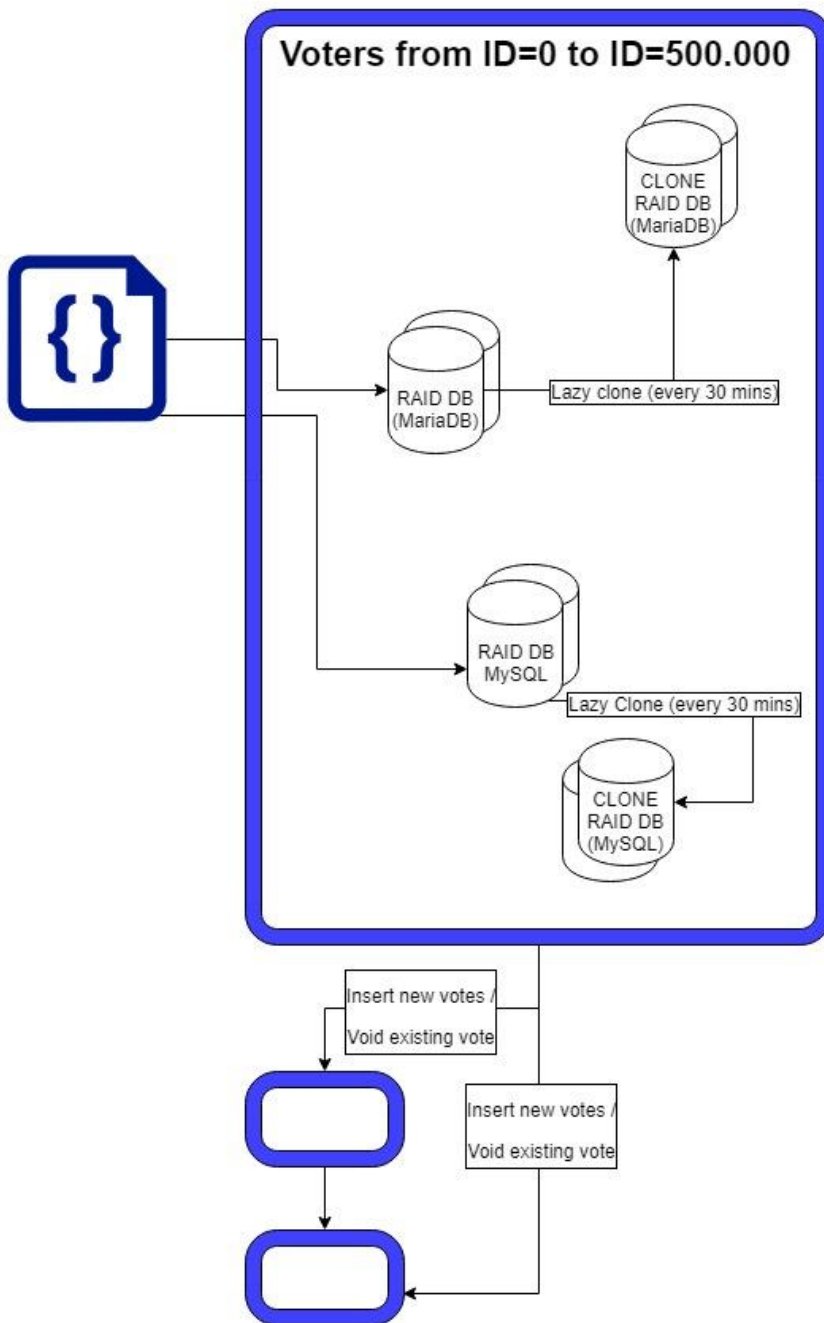


Image 3. Data storage process

## 9.7 From Legal Perspective

System supports both – public and secret voting. For secret voting, we separate voter id (respondent id) from votes (responses) table, and use unique hash instead. That hash is used to check if vote exists by voter, and only he can generate his has from his electronic signature.

## 9.8 From Evolution Perspective

Database is BCNF normalization form, which means that if the scale is needed we can link the data via unique id to new tables, or by adding the new columns

## 9.9 From Performance & Scalability Perspective

The system is based on Relational Database Management System (RDBMS) and uses MySQL or MariaDB database. To address scalability - the data is separated:

- To address scalability, and to not run SQL query of full data set the tables are separated:
  - ID-wise: to different servers by ID's of guest-only, and by voters by 500,000
- To avoid database locking for writing, database is also have separation:
  - Table-wise for specific tables:
    - Logs table sits in a separate database

## 9.10 From Security Perspective

To address scalability - the data is separated:

- To protect from wide-attack on full voters base the data is separated:
  - ID-wise: to different servers by ID's of guest-only, and by voters by 500,000
    - Counters table is attached to the same database structure – guests (IP-based – for anti-brute-force and anti-spam protection), and voters by 500,000.
  - Table-wise for specific tables:
    - Logs table sits in a separate database
- Each database row is written into two different database software (MySQL and MariaDB) at the same time.
  - Exploits for MySQL probably won't work for MariaDB.
- Database rack in data center 1 connected to other two DB racks in other datacenters via different internet lines on three different physical locations (three different cities).
  - Remote / physical hack to one data-center probably won't impact data in others, as the others only supporting voiding and inserting.
- **System-halt on change**
  - Other two servers always compare data for changes in server 1, and if un-allowed change is detected the system, the whole system is halted for writing until it is approved by admin.
- Trusted data-encryption
  - Voter-generated hash in secret-vote table is encrypted with **sha2-384** algorithm.

## 9.11 From Availability & Resilience (Recovery) Perspective

As it RDBMS MySQL/MariaDB with ID-wise & table-wise separation.

- Each database row is written into two different database software (MySQL and MariaDB) at the same time.
- Database rack in data center 1 connected to other two DB racks in other datacenters via different internet lines on three different physical locations (three different cities).

Each server is RAID based with dual write – meaning there is always two exact SSD disks in each server.

Each database has a lazy-clone cronjob, running every 30 minutes.

# 10 Concurrency View

## 10.1 Process model (Performance & Scalability perspective)

The system is using concurrent process model. The “stoppie” is database write.

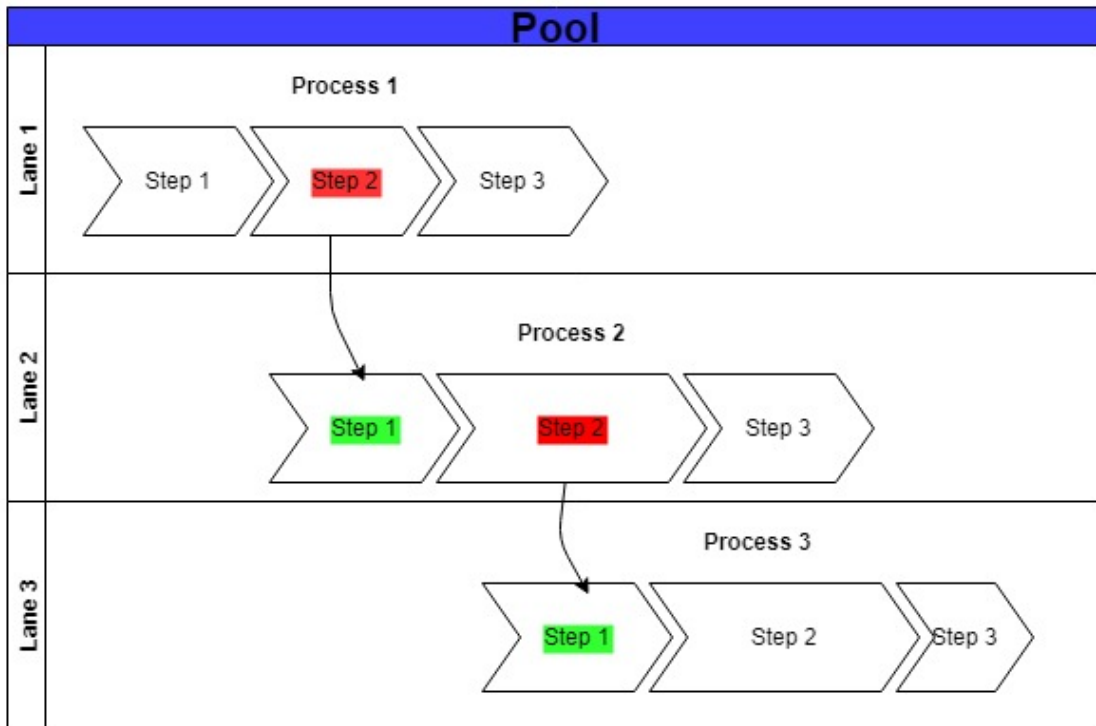


Image 4. Concurrent process























































## 10.2 Principles used from Performance & Scalability perspective

- System uses concurrent process model.
  - The “stoppie” is database write.

# 11 Development View

---

## 11.1 System models and main interfaces

- >  AgeGroup
- >  Assistant
- >  Cache
- >  City
- >  Configuration
- >  Coordinate
- >  Counter
- >  Country
- >  DecisionMaker
- >  Detect
- >  Entry
- >  FAQ
- >  File
- >  Formatting
- >  Import
- >  Install
- >  Language
- >  License
- >  Load
- >  Log
- >  Manager
- >  Message
- >  Node
- >  Notification
- >  Order
- >  Partner
- >  PostType
- >  Respondent
- >  Routing
- >  Search
- >  Semver
- >  Settings
- >  SpecialStatus
- >  State
- >  Status
- >  Style
- >  Taxonomy
- >  Update
- >  UseCase
- >  User
- >  Validation
- >  ZIP\_Code
- >  AbstractStack.php
- >  AbstractTable.php
- >  ElementInterface.php
- >  index.html
- >  ObserverInterface.php
- >  PartnershipInterface.php
- >  PrimitiveElementInterface.php
- >  PrimitiveObserverInterface.php
- >  RoleInterface.php
- >  StackInterface.php
- >  TableInterface.php
- >  WPUserInterface.php

## 11.2 Class diagrams

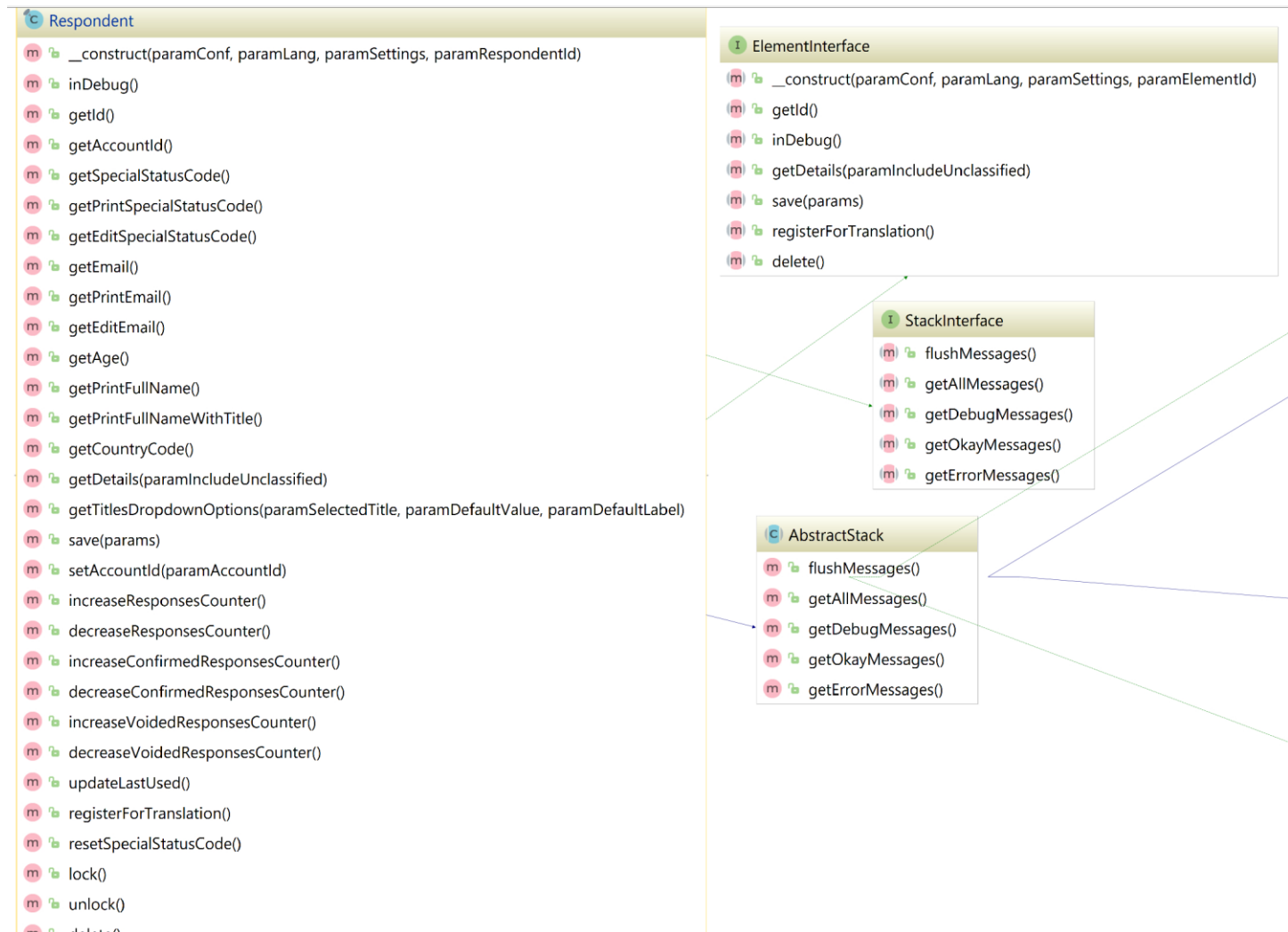


Image 5. Respondent model class diagrams – part 1

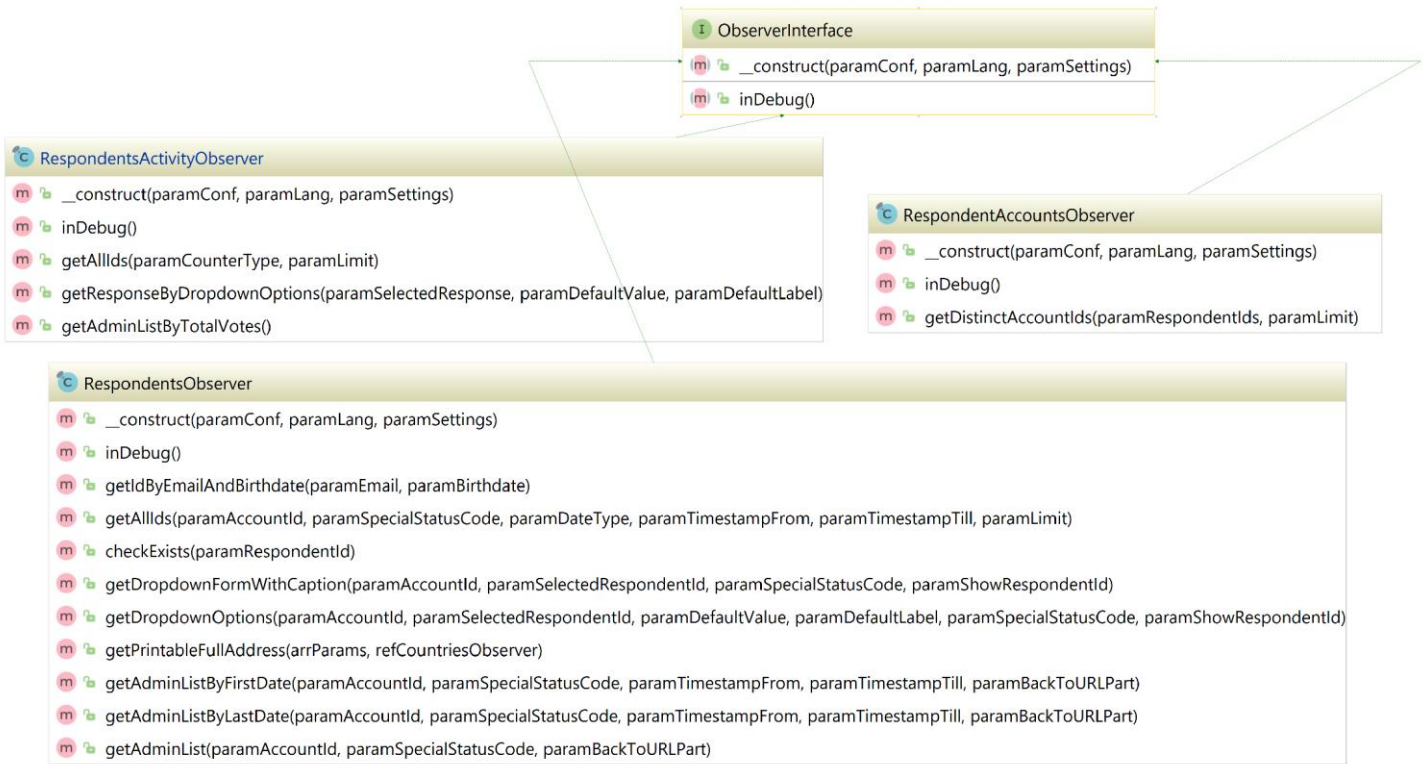


Image 6. Respondent model class diagrams – part 2

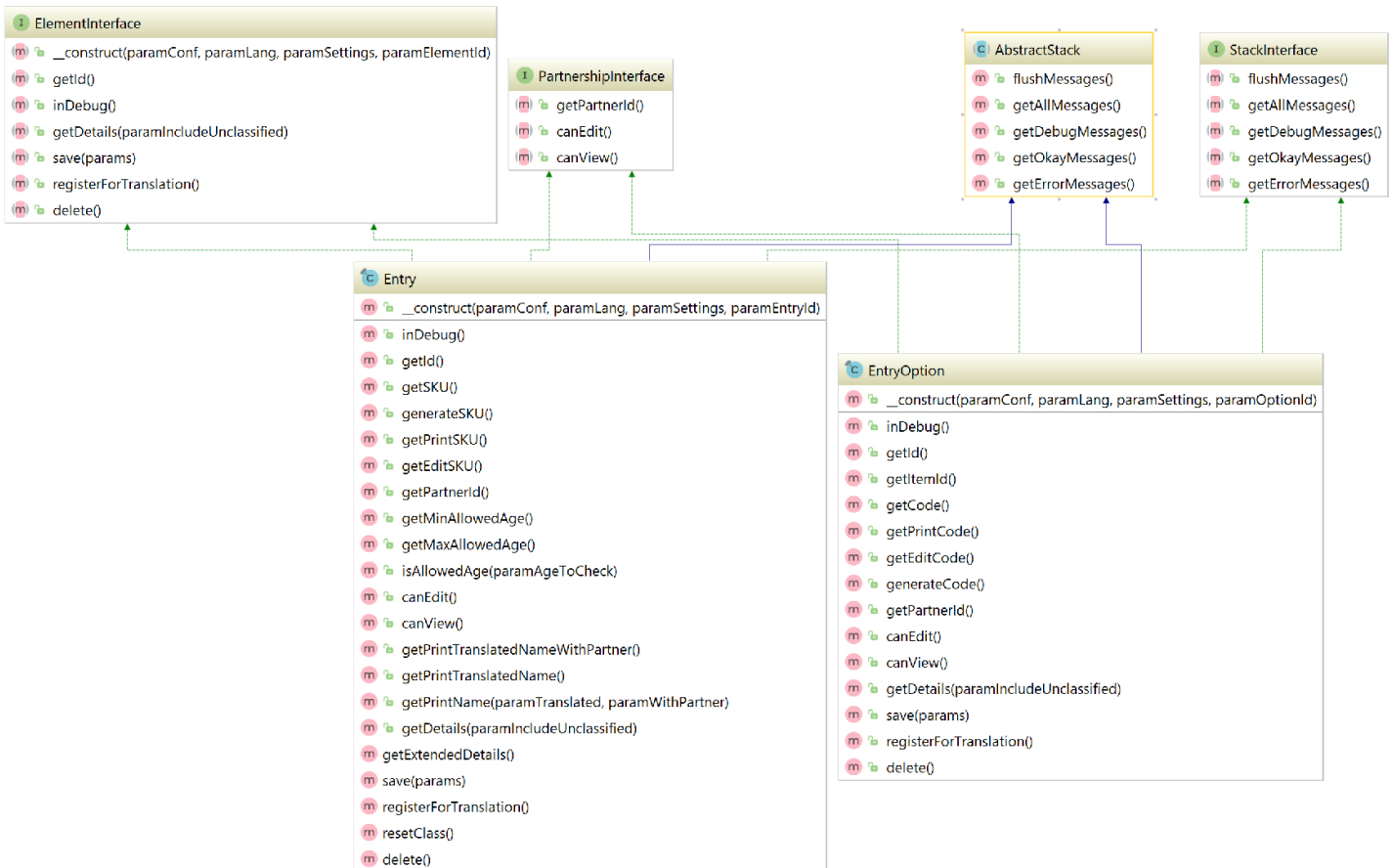


Image 7. Entry model class diagrams – part 1



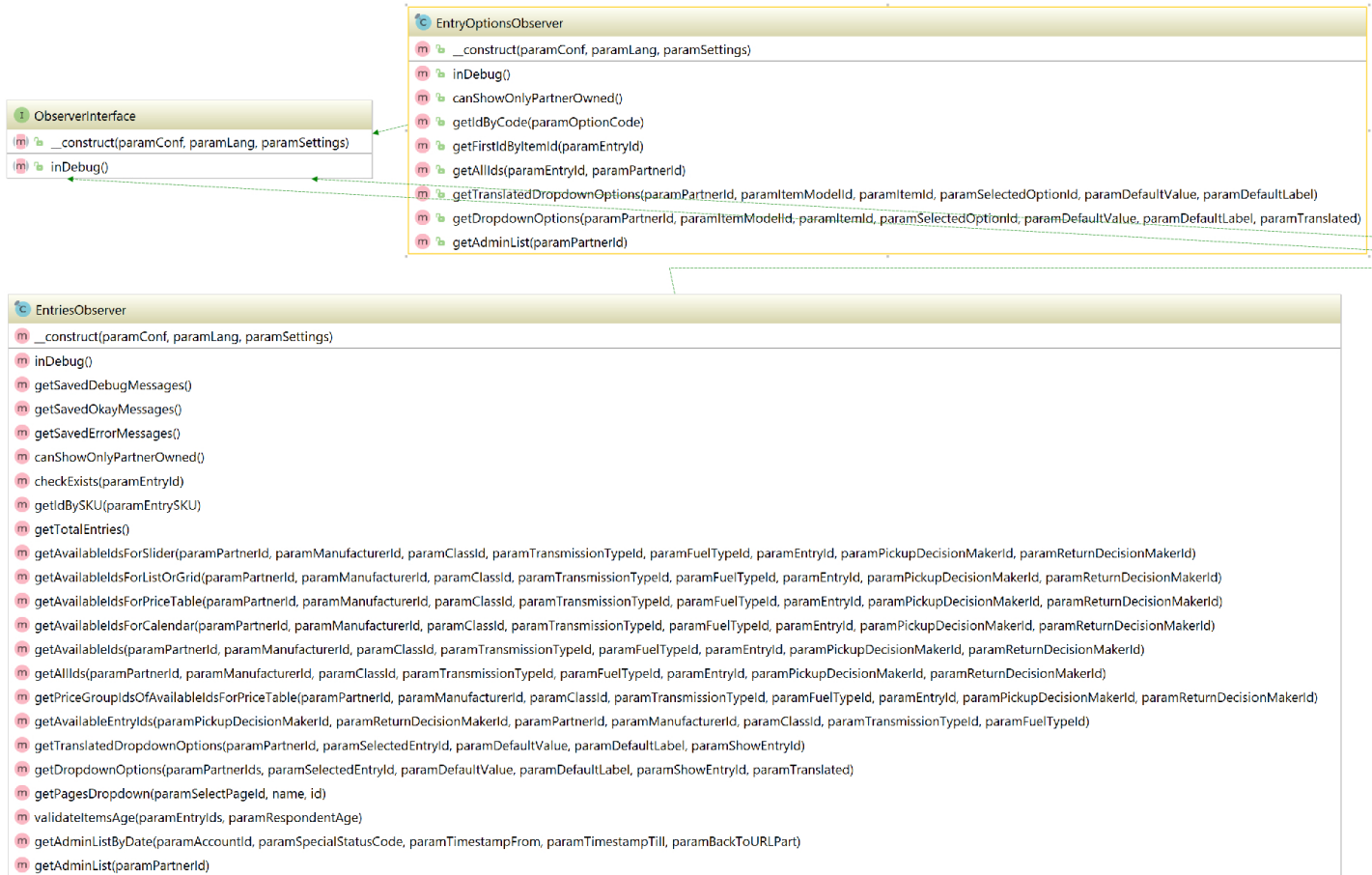


Image 8. Entry model class diagrams – part 2

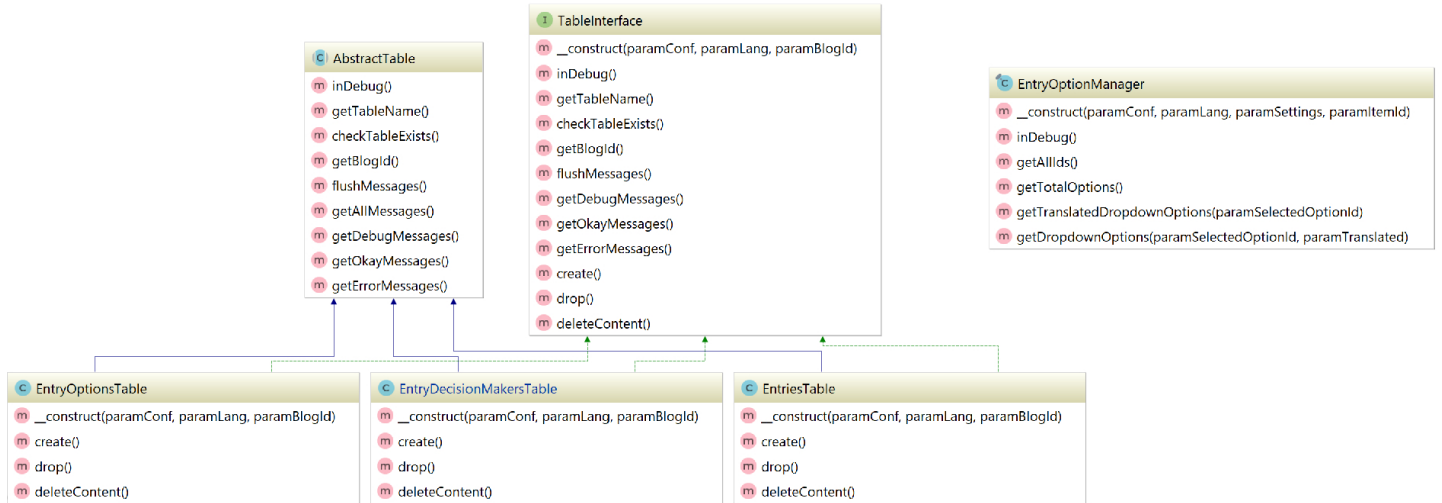


Image 9. Entry model class diagrams – part 3

### 11.3 S.O.L.I.D. MVC - Architectural Decision for Development View & Information Views

This architectural decision impacts information view, due to 1 model's element -> 1 table relationship.

1. S.O.L.I.D. MVC
  - 1.1. Status: accepted
  - 1.2. Deciders: CEO, Software Architect
  - 1.3. Date: Dec 10, 2018
2. Context and Problem Statement
3. Decision Drivers
4. Considered Options
5. Decision Outcome
6. Pros and Cons of the Options

### 11.4 SemVer - Architectural Decision for Development & Information Views

This architectural decision impacts information view, as the semantic version is stored in database.

1. Semantic Versioning (SemVer)
  - 1.1. Status: accepted
  - 1.2. Deciders: CEO, Software Architect
  - 1.3. Date: Dec 10, 2018
2. Context and Problem Statement
  - 2.1. Dependency hell when trying to activate the system, update the system, run beta versions, or process rollback.
3. Decision Drivers
  - 3.1. Known pattern to control the versioning.
  - 3.2. Widely-accepted
  - 3.3. Cross-platform, language-independent.
4. Considered Options
  - 4.1. My own versioning system (A.B)
  - 4.2. SemVer
5. Decision Outcome

Chose option 2 (SemVer).

  - 5.1. Positive outcome:
    - 5.1.1. Simple upgrades, rollbacks, version comparing – we always know which version is later, which is earlier.
  - 5.2. Negative outcome:
    - 5.2.1. A bit more of work to implement
    - 5.2.2. Not that simple versioning
6. Pros and Cons of the Options
  - 6.1.1. Option 1 (My own versioning system)

Pros: Code already exist. Can be used from start.

Cons: No way to work with other system, no easy way to notify correctly upgrade service.

No support for patching.

6.1.2. Option 2 (SemVer)

Pros: Cross-platform, language-independent, widely accepted.

Cons: Many params on version, not that simple versioning.

# 12 Deployment View

## 12.1 Network & servers infrastructure

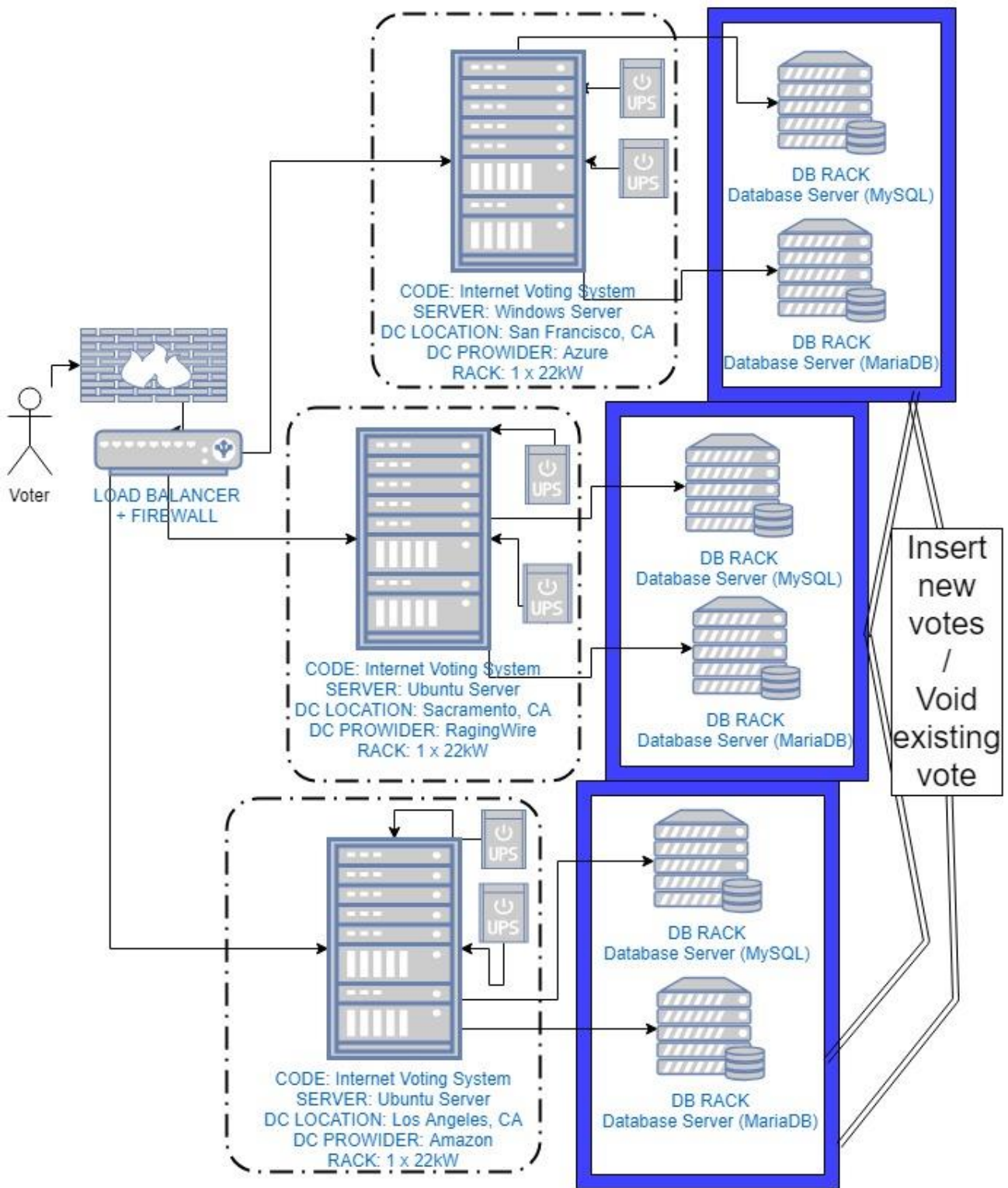


Image 10. Network structure

## 12.2 Principles used from Security perspective

- All remote data has been sent via secured HTTPS network with valid certificate of Trusted CA (Certification Authority) – a root CA.
- All internal data processes (cloning) is separated via firewall from external access and goes only via internal network or via external network, but with only exactly approved IP's of those servers.

## 12.3 Principles used from Legal perspective

- All data centers must be located in the same country where the elections will happen.
- Any 3<sup>rd</sup> party software should be made by friendly country to the voting country, i.e. for Lithuania or US that would be a country that is a member of NATO or EEA (European Economic Area, which includes EU member-states and Switzerland).

# 13 Operational View

---

## 13.1 Stakeholders

- ✓ Network (server) administrators
- ✓ Software architects
- ✓ CEO

## 13.2 Principles used from Performance & Scalability perspective

Performance monitoring via performance counters – an on/off screen output of how many milliseconds it took to load each website section, and how many overall queries we ran. This information can be logged to file if total load was longer than 5 seconds, and will include all SQL queries that took longer than 1 second.

## 13.3 Principles used from Availability & Resilience (Recovery) perspective

- **Functional migration:**
  - **Parallel functional migration** – the code is in servers located in 3 different data centers, there is always two simultaneous connections to databases (MySQL and MariaDB) and each database connection is scope-based (for guests, for voters with ID range 0-499,999, 500,000-999,999 etc.), so the functional migration first will be done for 1 datacenter for 1 database type (i.e. MariaDB) and for a scope of voters (i.e. 0-499,999). And if it will be ok there, we will go to next server.
- **Data migration:**
  - **Auto roll forward, manual rollback data migration** – the code have a SQL upgrade scripts for database that will run after “Upgrade” button is clicked in site administration for the internet voting system. The system administrator will have to make a “frozen” backup before the upgrade of current database and current uploaded images (/wp-content/uploads/InternetVotingSystem/).
- **Installation and upgrade:**
  - **Semi-automatic code pull from private GitHub repository** – the pull has to be done by admin after he logins to main server (to upgrade all servers) or to child server (to upgrade one server).
  - **Manual rollback** – the pull of previous version ‘tag’ in GitHub has to be done by admin after he logins to main server (to upgrade all servers) or to child server (to upgrade one server).
- **Backup and restore:**
  - 3-2-1 backup model (3 copies - 2 in local HDD’s & 1 in the cloud)
  - And 20-30 steps handbooks for all main failure scenarios (5-10 scenarios, i.e. “database raid disk fault, and replacement of the disk”).
  - Experiments in production with database is down (Chaos engineering).
- **Support:**
  - 24/7 support via e-mail with at least 1 technical specialist (server administrator) online. Totally 4 full-time employees will be needed for this.
  - Technical specialist phone number will be known to management of main customer (i.e. “by San Francisco State’s Government’s Head of IT”)
  - 24/7 support via phone & e-mail of 2 non-technical specialists. Totally 8 full time employees will be needed for this.

- **Alerting:**
  - “Uptime Robot” ( <https://uptimerobot.com/> ) software will do pings to servers.
  - The e-mails has to be send if server is down, got back up. The e-mail has to be configured by technical administrator to receive by his phone every 5 minutes. Also the “iPad” application that will send a sound on down has to be installed on technical administrators “iPad” and notifications & notification sound must never be disabled for this application if he is on-duty at that time (during his working hours). On non-working hours only the sound can be disabled, notification still has to pop-up in the screen.

#### 13.4 Principles used from Security Perspective

- ✓ **Logging & Monitoring:**
  - Separated logging and monitoring systems

#### 13.5 Principles used from Legal Perspective

- ✓ All support staff (technical & non-technical) has to be citizens of internet voting system installation country.
- ✓ All support staff must a security-clearance and government permit to work with secret information.