

Invibox

Technical Specifications
Blue Fountain Media

Revisions

Version	Author	Date	Notes
0.1	Nuno Pinto	09/03/2013	First draft
0.2	Nuno Pinto	09/04/2013	Added stack diagram
0.3	Nuno Pinto	09/05/2013	Added user database model
1.0	Nuno Pinto	09/05/2013	First version

Table of Contents

REVISIONS 2

1. OVERVIEW..... 4

2. TECHNICAL STACK 5

 2.1. COMPONENTS DIAGRAM..... 5

 2.2. DESCRIPTION 6

3. DATA CLASSES 9

 3.1. USER DATA MODEL..... 9

 3.2. DESCRIPTION 10

 3.2.1. *User Profile*..... 10

 3.2.2. *External Connections*..... 10

 3.2.3. *Email*..... 10

 3.2.4. *Events* 11

 3.2.5. *Contacts*..... 11

 3.2.6. *Groups Settings*..... 11

 3.2.7. *Applications* 12

 3.2.8. *Other* 12

4. BACKEND ARCHITECTURE..... 13

 4.1. OVERVIEW..... 13

 4.2. IMPORTANT RESOURCES 13

 4.3. SECURITY..... 14

5. FRONTEND ARCHITECTURE..... 15

 5.1. OVERVIEW..... 15

6. LICENSE GLOSSARY 16

1. Overview

This document's intent is to expose the application Invibox from a technical point of view.

The whole of the application is in fact a complex system, composed by several layers of technology, working together to make the Invibox experience possible.

Invibox is both a mail server and client, with particular features that make the system unique. It starts on the mail server, with custom filters to identify unwanted email messages, and ends on a flexible and intuitive browser application that presents a set of features to the User that goes beyond email, such as chat, calendar, to-do list and media library.

Having in mind the projected growth of the user base, the choice of technologies and the architecture of the application were meant to cope with three premises:

1. The system has to be scalable;

The hardware needs of the system will increase and it's important that the various components can work on multiple machines flawlessly. It is also important that the server software, including backend, can function with different frontends on the future (for instance, mobile app, tablets or desktop apps). This is the main reason why backend is detached from frontend.

2. The system has to rely on open source, industry standard and well-maintained technologies;

Closed software can be discontinued and thus break an entire system; open source don't. The chosen software for the different parts of the system is open source, maintained by large communities and, in most cases, with great companies backing up.

3. The system has to be secure.

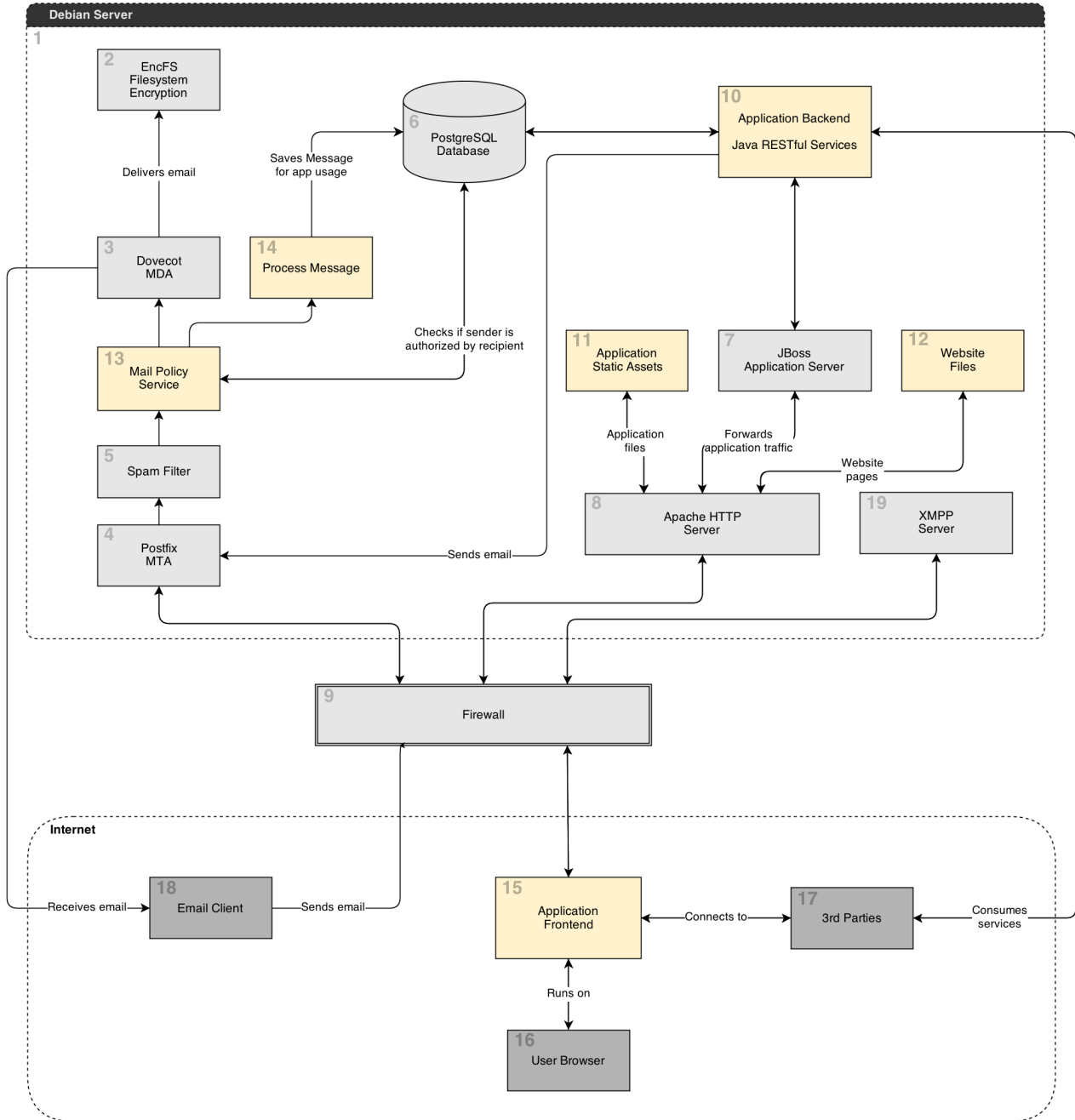
Every internet-facing application should be secure; but, when dealing with user sensible data, such as emails, security is by design. Features should not compromise security, but the opposite.

This document structure reflects vaguely that from the application. It is divided into four main sections that explore four different parts:

1. The stack of technologies behind Invibox and how they interact;
2. The data classes presented through a database diagram;
3. The architecture of the backend application;
4. The architecture of the frontend application.

2. Technical Stack

2.1. Components Diagram



KEY

Light gray	Technologies and software in which the application will rely on and that will require custom configuration.
Dark gray	3rd party tools or services that cannot be controlled
Yellow	Custom developments by BFM

Figure 1 - Components diagram

The diagram above shows the different technologies and how they interact. The light gray boxes, despite representing commercial tools, require deep configuration to work together, whilst on the dark gray ones there is no control. The main focus, though, should be on the yellow one, as they represent the development product.

2.2. Description

1. Server Operating System: [Debian](#) Stable (codename “Wheezy”) v7
 - It’s free, mostly GNU GPL;
 - It’s a stable OS, widely used for web and email servers;
 - Has a great amount of available software ready to install on its own repositories;
 - Has a great community of supporters and users;
 - Is supported by HP and other big names of the industry.
2. File system Encryption: [EncFS](#)
 - Encrypts all mail saved on the server;
 - GPL license;
 - Has Java support.
3. Mail Delivery Agent (MDA): [Dovecot](#)
 - Provides IMAP and POP support;
 - It is know for being secure;
 - Is well supported by Debian;
 - Complete IMAP4rev1 and POP3, IPv6, SSL and TLS support;
 - MIT and LGPL licenses.
4. Message Transfer Agent (MTA): [Postfix](#)
 - Provides SMTP support;
 - IBM Public License;
 - It’s a standard mail tool in the Unix world.
5. Spam Filter: [Spamassassin](#)
 - Apache license;
 - Checks all mail for SPAM signatures before it enters the Invibox flow;
 - Apache license;
 - Integration with Postfix.
6. Database Engine: [PostgreSQL](#)
 - Postgre License;
 - Has a solid feature set;
 - Industry standard for robustness, performance and availability.
7. Application Server: [JBoss AS](#)
 - LGPL;
 - Powers Java applications;
 - Is a full powered Java EE (Enterprise Edition) application server;

- JBoss has both a free edition (called AS) and a commercial one (EAP);
 - JBoss EAP offers full support;
 - In the case that support is needed, Invibox can upgrade service without changing server.

8. HTTP Server: [Apache HTTP Server](#)

- Apache license;
- Forwards traffic to JBoss;
- Serves static assets, such as Javascript and Stylesheets to the Frontend application;
- Serves static or dynamic web pages.

9. Firewall

- Dedicated server running a firewall to analyze incoming requests and reject any suspicious connections.

10. Application Backend

- Written in Java;
- Framework Jersey, an API that implements JAX-RS (Java API for Restful Web Services);
 - License is CDDL and GPL;
- The backend application will provide a REST interface for the frontend application and for 3rd party programmers developing Invibox apps.

11. Application Static Assets: Serves frontend application

- HTML;
- Javascript libraries and custom code;
- Stylesheets (CSS);
- Images;
- Videos;
- Cached content

12. Website Files

- Static or dynamic content for the Invibox website and CMS.

13. Mail Policy Service

- Custom filter written by BFM that connects to Postfix;
- Checks if a mail message sender is allowed by the recipient;
- Uses [Postfix Policy Server](#) to reject unwanted email messages directly on the server;
- Written in Java.

14. Process Message

- Acts when a message is allowed to be delivered to an Invibox recipient;
- Saves the message and associated metadata on the database for in-app usage;
- Written in Java.

15. Application Frontend

- “A Single Page” Javascript application;
- Connects to backend using AJAX calls to a Restful API;
- Uses framework [AngularJS](#) (MIT License);
- Uses other libraries such as jQuery for animations and DOM manipulation (MIT License).

16. User Browser

- It’s where the frontend application runs;
- Invibox will use technology available on modern browsers, such as CSS 3 properties, which might not be available on older browsers.
 - There is no plan to support outdated browsers.

17. 3rd Parties

- External networks connected to Invibox;
 - Google Mail and Calendar, Yahoo, Facebook, etc.
- Invibox is constrained by
 - Users permissions to use these networks;
 - External API availability and limitations.

18. Email Clients

- A user can set up his Invibox account on his email client of choice;
 - Outlook, Thunderbird, Mail, etc.
- Invibox provides SMTP and IMAP availability;
- User is responsible for the use of external software connected to Invibox.

19. XMPP Server: [Prosody](#)

- BSD license;
- XMPP is an open protocol for chat applications;
- XMPP server provides the built in chat functionality.

3. Data Classes

3.1. User Data Model

The diagram bellow shows the User-centered database design. Because of its size, the picture had to be rescaled to fit the page. For a full and interactive version of the database model, tables, fields and its inner relationships, please visit <http://invibox-database-model.bfmdev9.com>.

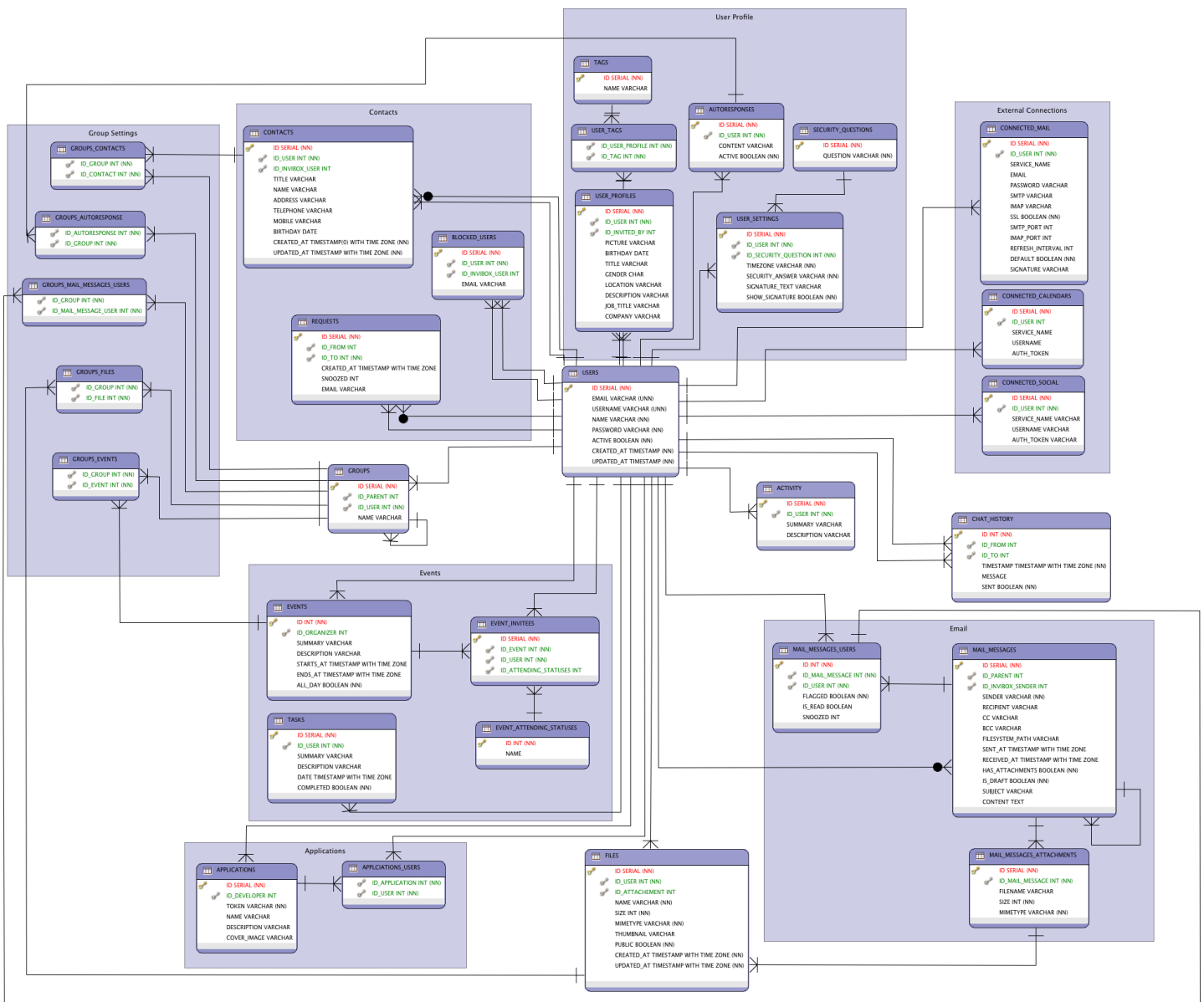


Figure 2 - User Data Model

3.2. Description

3.2.1. User Profile

Users

Essential User information, such as user name and password;

User Profiles

Profile information for the users, including a key to the user by whom he was invited.

User Settings

User personal preferences. Includes a key to one of the preset Security Questions.

Security Questions

List of security questions for the user to choose from during signup.

Tags

List of tags preloaded by Invibox and added by the users.

User Tags

Relates a User to the tags he added on his profile. When a user enters a tag that is not in the Tags table yet, it is added automatically.

Auto-responses

Contains auto-responses created by users.

3.2.2. External Connections

Connected Mail

List of connected mail services authorized by the user or credentials for custom servers.

Connected Calendars

List of external calendar services authorized by the user.

Connected Social

List of social network sites authorized by the user to be accessed by Invibox.

3.2.3. Email

Mail Messages

Mail sent or received; if sent, must contain an Id from an Invibox user. Can reference itself to represent replies and mail chains.

Mail Messages Users

Relates a message with sender and recipients. Has representations for statuses Read, Snoozed and Flagged per message per user.

Mail Messages Attachments

List of attachments per message.

3.2.4.Events

Events

Event data, with reference to an Invibox user organizer.

Event Invitees

List of other users invited to an event and their statuses.

Event Attending Statuses

List of possible values: Attending, Not attending, Maybe.

Tasks

List of tasks from a user.

3.2.5.Contacts

Contacts

List of contacts from an Invibox user. Contacts can have an Invibox account or not.

Requests

List of current requests to a user. Requests may be from an Invibox user or from outside.

Blocked Users

List of blocked users. User may be blocked by an Invibox Id or email address.

3.2.6.Groups Settings

Groups

List of groups created by the User. It is preloaded with the default groups from Invibox. The reference to itself represents a Group within a Group.

Groups / Contacts

Defines relationship between user contacts and groups.

Groups / Mail Message Users

Defines relationship between user mail message and groups.

Groups / Files

Defines relationship between user files and groups.

Groups / Auto-response

Defines which groups should see which auto-response.

Groups / Events

Defines relationship between user events and groups.

3.2.7.Applications

Applications

List of Invibox applications / widgets.

Applications Users

List of applications that a user as installed.

3.2.8.Other

Files

Files relate to an attachment that is saved in the Library. It is private by default but can be public if the user wants to share.

Activity

Activity history from the users.

Chat History

Saved history from Invibox chat.

4. Backend Architecture

4.1. Overview

The Backend for Invibox is a Java application that relies on the following specifications:

1. Java Persistence API (JPA)
It is meant to describe the management of relational data; provides mapping between Java classes (called persistence entities) and the database tables.
2. Java API for RESTful Services (JAX-RS API)
Provides support for creating web services according to the REST architectural pattern. In a REST architecture, the client (frontend) makes a HTTP request to the server (backend) where it defines the resource and action to perform; for instance, GET USER #1.
The Jersey Framework, to be used, is the de facto JAX-RS implementation, from Sun.

Therefore, the backend application is meant to be a central piece that manages everything that a User can do on the frontend. It accepts client requests, processes data from the database and responds to the request.

4.2. Important Resources

Everything in a RESTful server is a resource. These resources are meant to interact with objects that, in the case of Invibox, are stored by the database. Every resource allows for retrieve, create, modify or delete a data entry. So the Invibox backend will need to expose as many resources as the objects that the User can interact with.

The most important are:

- www.invibox.com/api/v1/users
- www.invibox.com/api/v1/profile
- www.invibox.com/api/v1/settings
- www.invibox.com/api/v1/contacts
- www.invibox.com/api/v1/events
- www.invibox.com/api/v1/todos
- www.invibox.com/api/v1/messages
- www.invibox.com/api/v1/groups
- www.invibox.com/api/v1/files
- www.invibox.com/api/v1/profile

- www.invibox.com/api/v1/applications
- www.invibox.com/api/v1/connected_email
- www.invibox.com/api/v1/connected_calendars
- www.invibox.com/api/v1/connected_social

4.3. Security

Security is also a concern of the backend. Users authenticate via a password encrypted using the SHA-512 encryption algorithm, with single use, user dependent salt.

After login, the backend is responsible for maintaining a session token and to validate requests against it, to prevent forgeries.

5. Frontend Architecture

5.1. Overview

The Frontend application is the Invibox face to the User. Therefore it has to be intuitive, fast, lightweight and robust. The trend on web applications is the Single Page Application (SPA) approach. This avoids page reloads, is cheaper to the server (in terms of number of requests) and mimics a desktop application experience.

Invibox frontend relies on the Angular JS framework, a tool created and maintained by Google (and a large open source community) to write SPA MVC applications.

The MVC architecture stands for Model-View-Controller and it's a paradigm to control the data flow of an application through the three essential concepts.

1. Model
Represents the data that has a known object structure, such as User or Message.
2. View
Is the markup that formats the data presentation on the page.
3. Controller
The controller is the core piece that binds Models to Views and where most actions are performed. It's also the controller that triggers requests to the REST server and retrieves and parses the response.

6. License Glossary

- Apache License: <http://www.apache.org/licenses/>
- BSD: http://en.wikipedia.org/wiki/BSD_licenses
- CDDL: <https://jersey.java.net/license.html#/cddl>
- GNU GPLv2: <http://www.gnu.org/licenses/gpl-2.0.html>
- IBM Public License: http://en.wikipedia.org/wiki/IBM_Public_License
- LGPL: http://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License
- MIT: <http://opensource.org/licenses/MIT>
- MySQL: <http://www.mysql.com/about/legal/licensing/index.html>
- PostgreSQL: <http://opensource.org/licenses/postgresql>