

# Application Security

*Léo Gabaix – Jean-Paul Morcos Doueihy – Francesco Mosanghini*

## Security by Design

### A free note-taking application

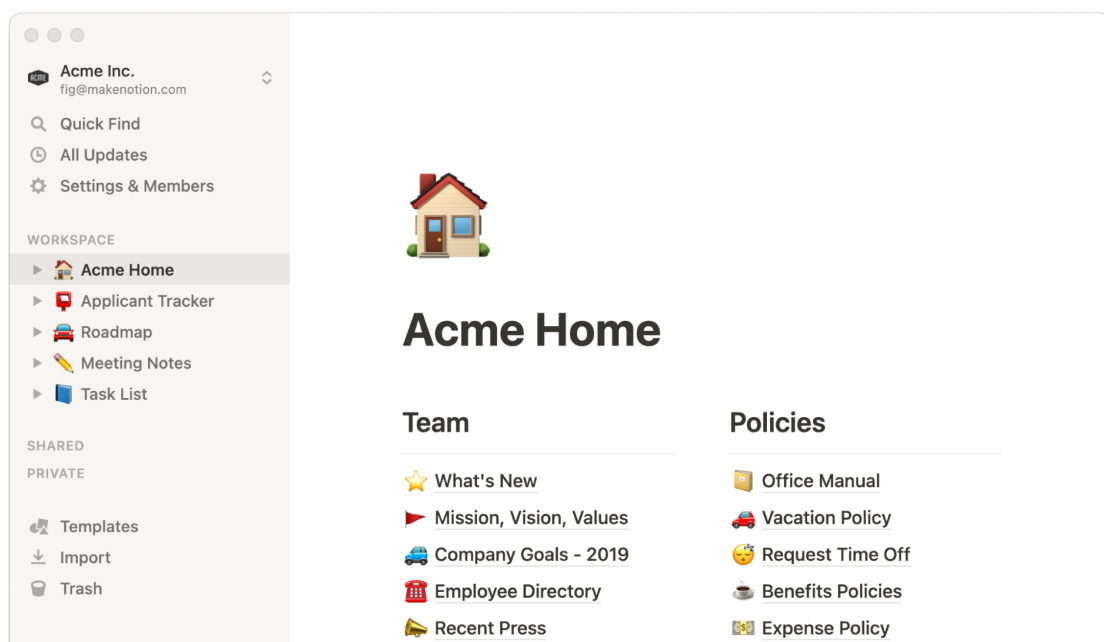
<b>1. Context.....</b>	<b>1</b>
<b>2. Development stack.....</b>	<b>1</b>
<b>3. Security by Design.....</b>	<b>2</b>
3.1. Security features.....	2
3.1.1. HTTPS.....	2
3.1.2. Password hashing.....	2
3.1.3. In-transit data encryption.....	2
3.1.4. Access Control.....	3
3.1.4.1. XACML.....	3
3.1.4.2. JWT.....	3
3.1.5. Secret Management.....	3
3.2. Secure development.....	4
3.2.1. Branch protection.....	4
3.2.2. Dependency check.....	4
3.2.3. Code Analysis.....	5
<b>4. Modelization.....</b>	<b>6</b>
4.1. Architecture scheme.....	6
4.2. Data representation.....	7
4.3. API routes.....	8
<b>5. Minimum Viable Product.....</b>	<b>9</b>
5.1. Must have.....	9
5.2. Should have.....	9
5.3. Could have.....	9
5.4. Will have.....	9
<b>6. Current status.....</b>	<b>10</b>

## 1. Context

Because we were tired of not finding the perfect collaborative free note-taking app we decided to create our own.

The idea is simple: as a user I can create and manage my notes but I can also create a collaboration group, invite some friends inside and start working together on shared notes.

The interface will be inspired by Notion, because it is very simple and intuitive:



*Notion's simple interface*

Our app will have a simple band on the side, containing the personal notes and the user's group(s). The main screen space will be used by the note editor.

## 2. Development stack

List of the chosen technologies to develop the app:

- **Frontend:** Vue.js
- **Backend:** Node.js + Express.js
- **Database:** MongoDB
- **Hosting server:** Linux (Ubuntu)
- **Version Control Source:** Git

- **Containerization:** Docker
- **Version Control Server:** Github

You can follow the project development using the following link:

<https://github.com/blueh0rse/note-taking-app>

### 3. Security by Design

#### 3.1. Security features

We will develop our app with the following security features in mind:

##### 3.1.1. HTTPS

The clients will communicate with our web server using HTTPS, guaranteeing the encryption of the traffic.

To do so we will issue a valid TLS certificate from a Certificate Authority (CA). When a client will visit our web application the user will see “🔒 **Connection secure**” on its browser.

Any request trying to communicate over HTTP will be automatically rejected by the server.

##### 3.1.2. Password hashing

We will of course not store the password in plaintext in our database. We will only store hashes.

To do so we will use the bcrypt hash function through the *bcrypt* javascript library.

This library is widely used because it is designed to be secure and slow (to prevent brute-force attacks), it also automatically handles the generation and storage of salts (salts are used to ensure 2 identical passwords generate a different hash), thus preventing rainbow table attacks.

Bcrypt is a trusted choice for this task.

##### 3.1.3. In-transit data encryption

It is crucial that the data transiting between the web server and the database is protected from any form of alteration. Its integrity must be our priority.

To achieve this protection we will issue another TLS certificate from a CA to ensure our web server is communicating over TLS with our database instance.

### **3.1.4. Access Control**

Because Broken Access Control is the number one risk for any web application, according to the OWASP Top 10, we must take security measures about it.

To address this risk we have two options:

#### **3.1.4.1. XACML**

XACML stands for eXtensible Access Control Markup Language, it is used to express and enforce access control policies. It is more often used in enterprise or in complex systems because it can define control policies in a very detailed and granular manner.

#### **3.1.4.2. JWT**

JSON Web Tokens are commonly used in authentication processes. A JWT encapsulates data about an authenticated user such as its identity and its role.

On one hand we can use XACML in addition to JWT. One will check user's authorizations whereas the other will manage the authentication process.

On the other hand, because XACML can be complex to implement in a short delay, we can use JWT for both authentication and authorization.

We will first try to implement XACML and if its complexity is too time-consuming we will go for JWT-only.

### **3.1.5. Secret Management**

In the fifth OWASP TOP 10 position there is the "Security Misconfiguration" risk. One common mistake is to hard-code an API key or a database password in a configuration file then commit the file to the version control server (Github, Gitlab...). If a secret is committed, everyone with read access to the repository can look at the file and see the line "API\_KEY=xxx".

To ensure our secrets will stay hidden we will use the *dotenv* node module. By using a '.env' file which always stays offline (out of the version control server) thanks to the .gitignore file, we are sure our secrets will never leave our computer and thus not leaking anywhere.

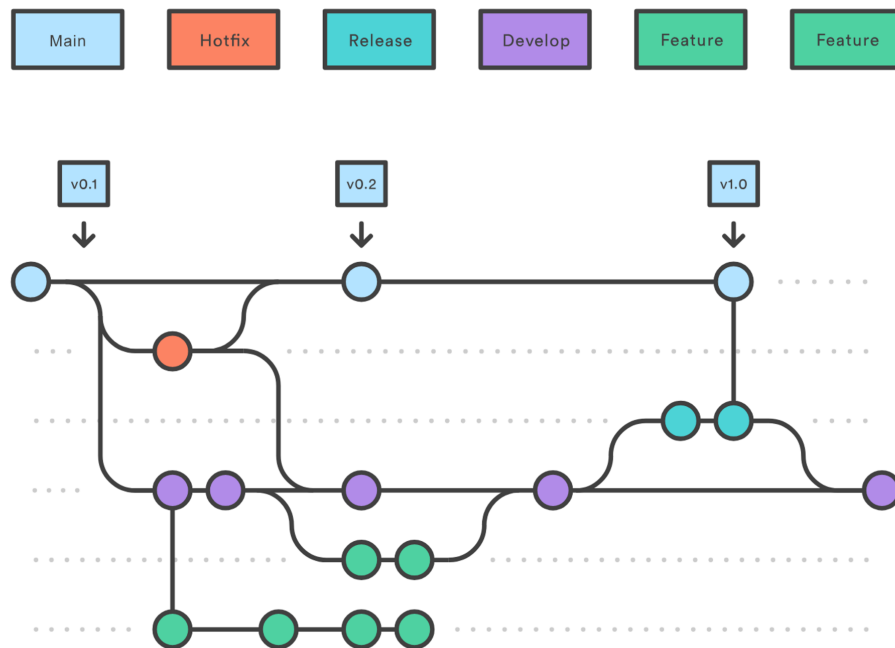
The "Secret detection" option is also set on the repository settings. It is an option directly available through Github. It allows Github to search for secrets (passwords, tokens...) automatically and send an alert if one is detected.

### 3.2. Secure development

Now that we described how our application will run securely we will talk about how we will maintain secure our code.

#### 3.2.1. Branch protection

To begin it is worth noticeable to say that we will organize our branches using the *Gitflow strategy*:



*a Git tree using Gitflow strategy*

This strategy was invented in 2010 and fastly became the most popular git strategy for development teams. It aims to considerably reduce branch merging conflicts.

Two branches are protected:

- *main*

Develop branch can only be merged into main though Merge Requests (MR) with the approval of two project members.

- *develop*

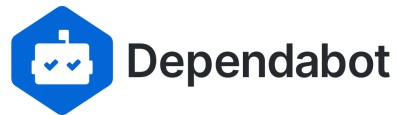
Features branches can only be merged into develop through MR with the approval of one project member.

#### 3.2.2. Dependency check

In the OWASP Top 10 sixth position there is the “Vulnerable and Outdated Components”. A project code can be secure but vulnerabilities can also be found in

the libraries used by the project. As nowadays 99.99% (to not say 100%) of the projects use external libraries it is a big risk to take into consideration.

Because keeping an eye on our dependencies is important and also because there are always too many dependencies the checking process must be automated.



On Github there is Dependabot to do the job. It scans dependencies listed in the configuration files (such as package-lock.json), checks for vulnerabilities and opens pull request (PR) to update a vulnerable dependency.

It is therefore activated on our repository.

### **3.2.3. Code Analysis**

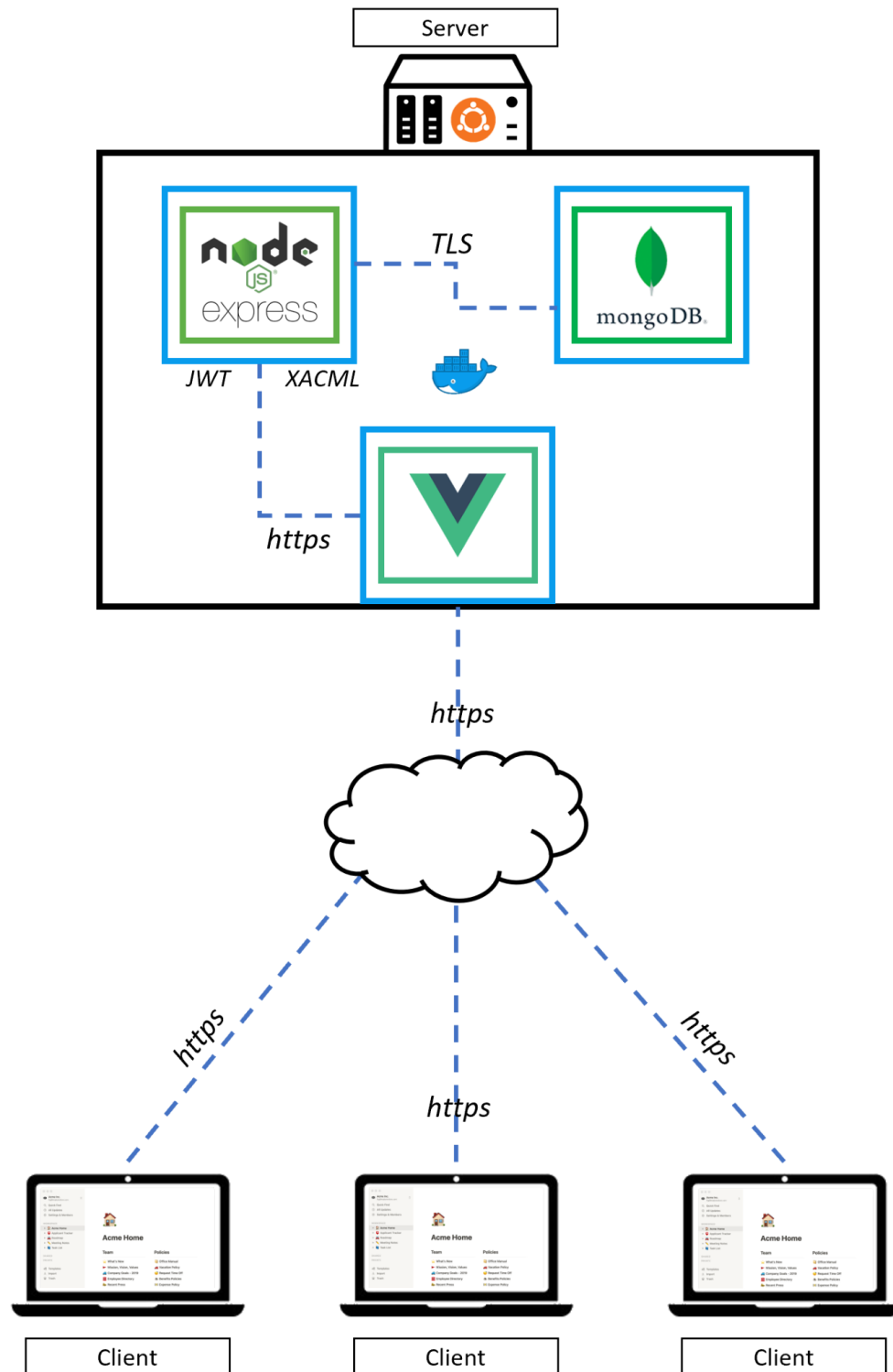
Because vulnerabilities can also come from our own code it is important to do security scans on it.

For this we will use Github Actions, a CI/CD platform integrated in Github to create workflows to automate actions like scanning our code.

We chose CodeQL to scan our code for security vulnerabilities at every commit because it is a good tool developed by Github, so perfectly integrated into the repository.

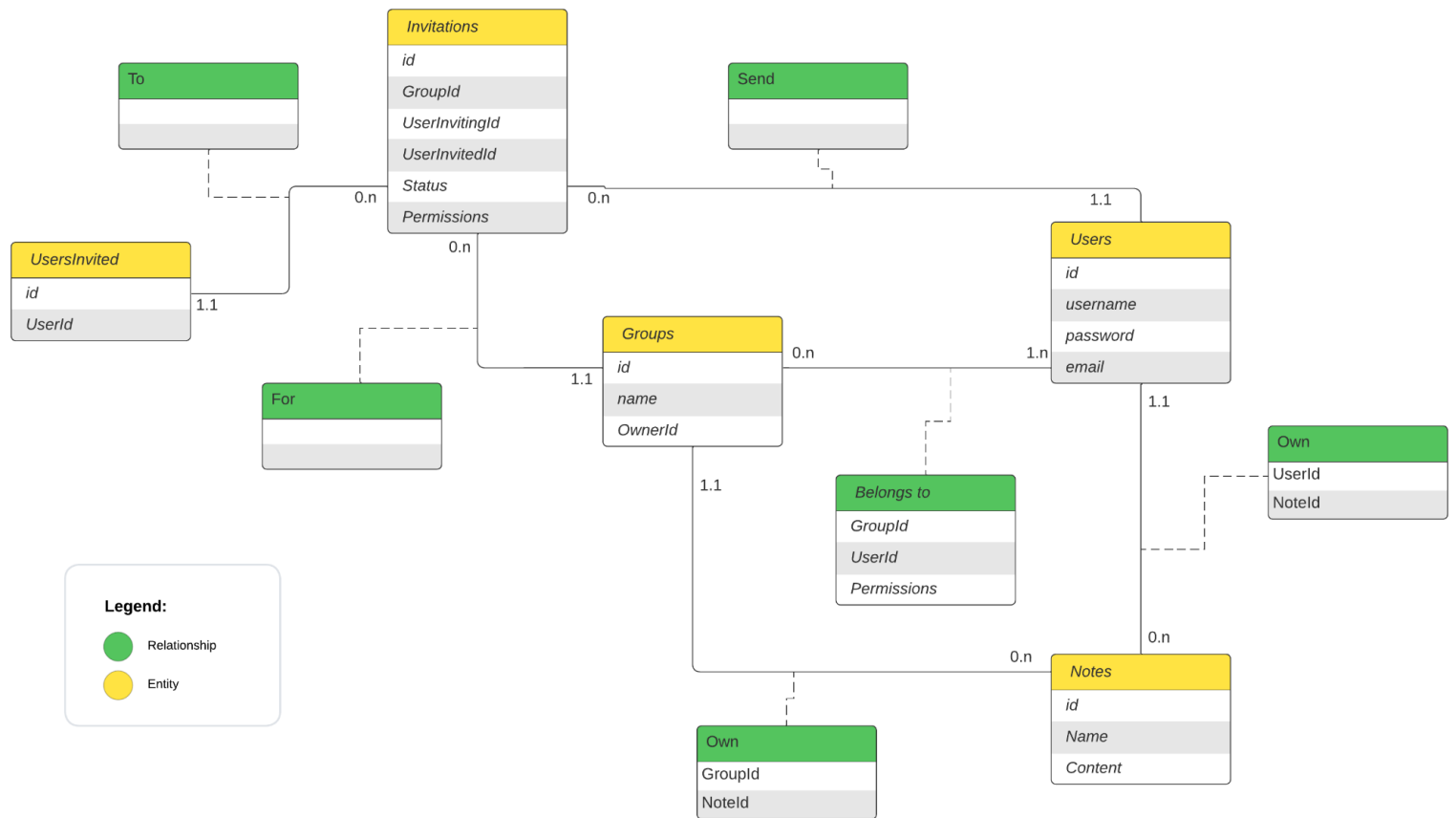
## 4. Modelization

### 4.1. Architecture scheme



*Architecture of our project*

## 4.2. Data representation



*UML representation of our database scheme*



### 4.3. API routes

Entity	HTTP Method	Endpoint	Action
Users	GET	/users	Retrieve a list of all users
Users	POST	/users	Create a new user
Users	GET	/users/{id}	Retrieve details of a specific user
Users	PUT	/users/{id}	Update a specific user
Users	DELETE	/users/{id}	Delete a specific user
Notes	GET	/notes	Retrieve a list of all notes
Notes	POST	/notes	Create a new note
Notes	GET	/notes/{id}	Retrieve details of a specific note
Notes	PUT	/notes/{id}	Update a specific note
Notes	DELETE	/notes/{id}	Delete a specific note
Groups	GET	/groups	Retrieve a list of all groups
Groups	POST	/groups	Create a new group
Groups	GET	/groups/{id}	Retrieve details of a specific group
Groups	PUT	/groups/{id}	Update a specific group
Groups	DELETE	/groups/{id}	Delete a specific group
Invitations	GET	/invitations	Retrieve a list of all invitations
Invitations	POST	/invitations	Create a new invitation
Invitations	GET	/invitations/{id}	Retrieve details of a specific invitation
Invitations	PUT	/invitations/{id}	Update a specific invitation
Invitations	DELETE	/invitations/{id}	Delete a specific invitation
Authentication	POST	/auth/login	Log in to a user account
Authentication	POST	/auth/register	Create a new user account
Authentication	POST	/auth/logout	Log out from a user account
Authentication	POST	/auth/refresh	Refresh access token

*Table containing all the API routes*

## **5. Minimum Viable Product**

To define our Minimum Viable Product (MVP) we will use the *MoSCoW* method.

Features are split into four categories depending on how important they are.

### **5.1. Must have**

- User can manage account
- User can manage notes

### **5.2. Should have**

- User can manage groups
- User can invite other users to group
- User can set permissions for group members

### **5.3. Could have**

- Users can collaborate in real time
- Notes can be exported as pdf






















### **5.4. Will have**

- Mobile application

## 6. Current status

Currently the application is in a very early state:

 **Blocked** /  **Doing** /  **Done** /  **Todo** /  **Backlog**

- **Environment setup**
  -  Repository
  -  Frontend
  -  Backend
  -  Database
  -  Containerization
- **Security features**
  -  HTTPS
  -  Password hashing
  -  In-transit data encryption
  -  Access control
- **Secure development**
  -  Secret management
  -  Branch protection
  -  Dependency check (Dependabot)
  -  Code security analysis (CodeQL)
- **Application features**
  -  Manage account
  -  Manage notes
  -  Manage groups
  -  Send invitations
  -  Set permissions
  -  Real-time collaboration
  -  PDF export
  -  Mobile application