Design Analysis

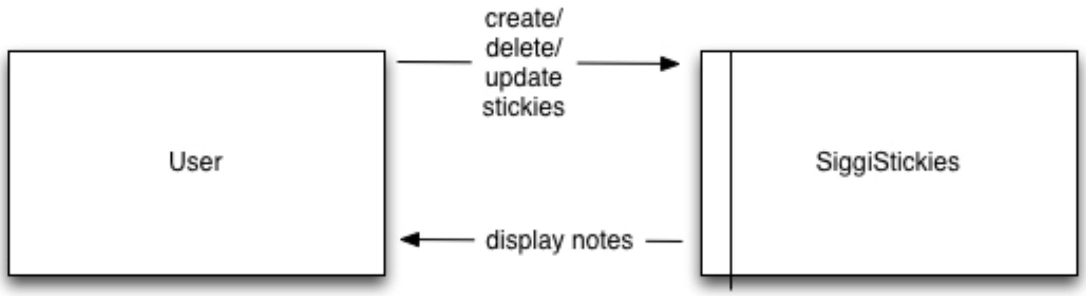
Overview

SiggiStickies is an application that emulates sticky notes such as post-its. A user places post-it like notes on his screen to store snippets of information.

Purpose and goals

The main goal of this project is to learn how to create a client-side application that does asynchronous calls to the server. I am also learning how to integrate a security model with such an application.

Context diagram

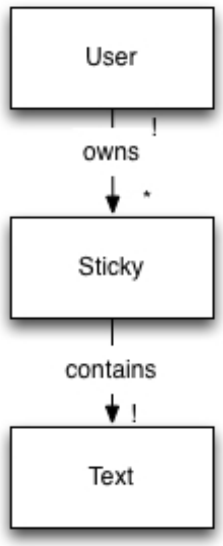


Concepts

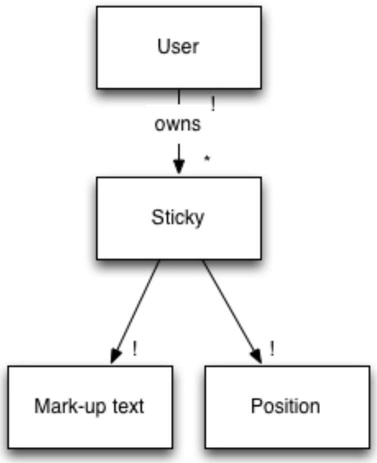
Key concepts

The key concepts are **user** and **sticky**. The user stores information about the user such as login credentials and name. The sticky stores the text contained in the actual sticky note and has a foreign key to a user.

Object model



EXTENSION:



Behavior

Feature descriptions

A user can:

- **create** a sticky
- **update** a sticky (by updating its text)
- **delete** a sticky

Moreover, if a user is logged in, stickies will persist across sessions.

EXTENSION:

A user can:

- **Bold**/underline/*italicize* text
- **Drag and drop** stickies

Security concerns

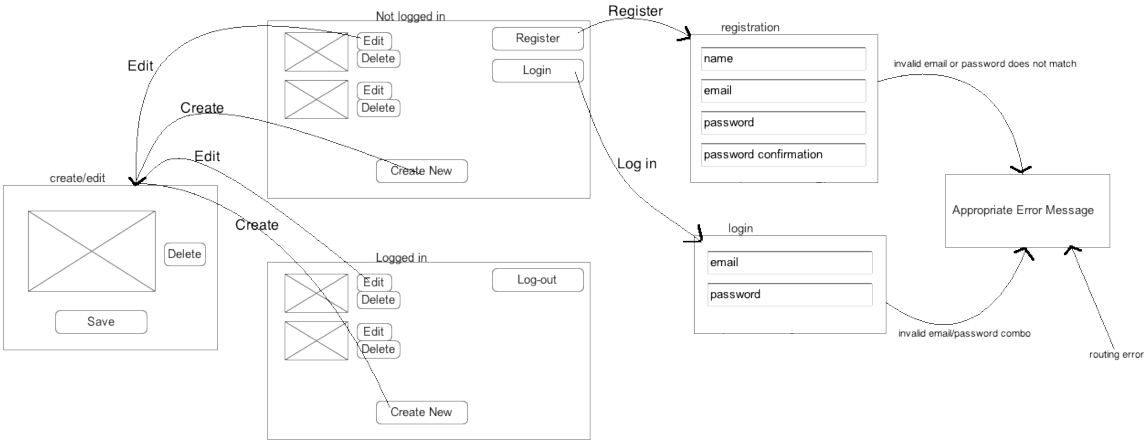
SiggiStickies has the following security requirement: It must acceptably guarantee that a user is only able to view/update/delete stickies that belong to him. To enforce this accounts will be password protected and each action on a sticky requires authorization.

The application could be vulnerable to the following kinds off attacks:

- User (inadvertently) might expose his password to someone else
- An attacker might alter his own cookie to gain access to restricted resources. This is mitigated by Ruby's cookie forgery protection
- An attacker might sniff out a cookie and extract password. This would be best mitigated by use of SSL for requests, but won't be implemented at this time
- An attacker might try to inject malicious text into sticky notes, which then are displayed on the application and might get executed as a malicious script. This is mitigated by sanitizing the text of a sticky.

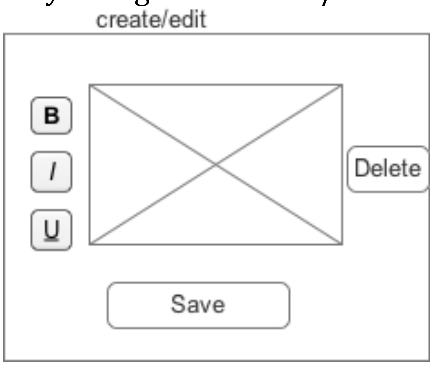
User interface

I use a rectangle with a cross in it as a placeholder for a sticky note. In the application the sticky will contain text.



EXTENSION:

Only change is in create/edit:



Challenges

Design challenges

When to require login:

1. Before using the application: This has high activation energy for the user, but is easiest to implement.
2. Before creating a sticky: This interrupts a user's workflow and is annoying.
3. Never require login (but keep the option if the user wants stickies to persist across sessions): This option requires the user to know that he needs to login if he wants his stickies to persist across sessions (he should be informed before unloading the page)

I decided to go with option three because it has the lowest activation energy and makes for the most enjoyable user experience.

Temporary stickies mergure:

Since I am not requiring login (see above) then what should happen to the "temporary" stickies when a user logs in:

1. Temporary stickies are cleared: This is clearly easiest to implement but is unexpected behavior and would be very frustrating for user.
2. Temporary stickies are merged with pre-existing: Expected behavior from a user's perspective and is unobtrusive.
3. User chooses whether he wants ALL temporary stickies to be merged: Similar to option 2 but with the added input of user.
4. User chooses whether he wants individual temporary stickies to be merged: Unnecessarily obtrusive.

I went with option 2 because it's what a user will expect and is the least obtrusive. I chose this over option 3 because if a user does not want a temporary sticky to be added he can always delete them (before or after signing in) so it would be unnecessary functionality

Maintaining stickies after log-out:

After a user logs out what should happen to his stickies:

1. All stickies remain in the view: This is not what a user would expect and introduces unnecessary security risks.
2. All stickies are removed from view: This is expected behavior from a user's perspective.
3. Only stickies that were previously temporary (i.e. those that were created before login) remain in the view: A plausible behavior but completely unnecessary. Moreover, after a user signs in he expects his stickies to only be visible during the session.

I decided to pick option 2 since others are unexpected and pose security threats.