BankChain - Product Planning

Chainable Technologies

H.G. van de Kuilen

rvandekuilen, 4226151

J.C. Kuijpers

jckuijpers, 4209915

E.J. Sennema

esennema, 4496582

M.R. Kok

mrkok, 4437659

I. Dijcks

idijcks, 4371151

May 2017

Contents

1   Introduction

Planning is important when doing a project, especially when working with a team, and with intermediate deadlines. Plannings gives insight in what is possible in a timeframe, and what is not. It also indicates when the project is behind a schedule.

This document focuses on the planning of the BankChain project of Chainable Technologies. It contains an explanation about the product, with a high level backlog explaining the features. Next, a roadmap is given for the duration of the project. Finally, a definition of a finished product is given.

2   Product

We will build an Android application that enables the user to verify that a public key belongs to a certain IBAN. The goal is to create more trust between two peers and possibly, verifiably, exchange their names. This is achieved by sending small amounts of money to each others bank accounts with an encrypted challenge and response. The product is a part of a larger project in the same context: to create a web of trust on Android phones.

2.1   High level backlog

There are a few elements of our application that we cannot do without. In this Chapter we will briefly discuss these elements. The elements are in order of priority.

Generate a private and public key

Our app has to generate a private and public key pair, to be able to encrypt and decrypt the messages send to a users bank account.

Banking API

Our app has to be able to interact with the API of a bank (Bunq in our case), to be able to transfer money and send messages between bank accounts.

Challenge Response

To confirm the public key and bank account of the user you want to validate, our app has to be able to send a challenge to another users bank account, and reply with a response that is a valid solution to the proposed challenge.

Output to Web of Trust

Once you have verified that a public key belongs to a bank account, the app has to store that result in the Web of Trust. That way the user doesn't have

to repeat the verification process. The user will see this in the app as a list of public keys that are trusted.

Intuitive UI

As IBAN numbers and public keys can become quite confusing numbers for the user, we will have to build an intuitive UI where the user has to deal with this as little as possible.

QR code

Since typing a public key is very error prone, the user should be able to scan a QR code to input the public key it wants to connect to an IBAN.

Integration with other teams

We will integrate our application with that of 2 other project groups. One group will build the web of trust, and the other will allow the user to expand this web of trust. We will combine these 3 applications into a singular product, that provides a web of trust, the ability to enlarge this web, and the ability to improve the trust within the web.

2.2   Roadmap

In sprint 7 and 8 we will have to integrate our app with the app of another project group. We will continue to discuss the integration with the other groups throughout the development.

Sprint 1

- Write the Architecture Design.
- Write the Product Planning.
- Write the Product Vision.

Sprint 2

- Implement a basic challenge/response system.
- Bank API integration (Bunq).
- Build a basic GUI.

Sprint 3

- Discuss integration with other project groups.
- Further improve our challenge/response system.

Sprint 4

- Create a Blockchain Interface.

- Improvement to the GUI.

Sprint 5

- Implement QR code scanning for exchanging public keys

Sprint 6

- Output the trusted public keys to the Web of Trust

- Show a list of trusted keys on the users request.

Sprint 7

- Integrate our app with other project groups.

Sprint 8

- Integrate our app with other project groups.

3   Product Backlog

We use MoSCoW to categorize our user stories.

"The MoSCoW method is an acronym commonly used to represent four hierarchical priority groups. Each requirement within a group shares the same priority." (Hatton, 2008). According to Hatton, the groups are as following:
Must have: The minimum functionality that the agent should have.
Should have: Advanced functionalities that would be nice to have if possible.
Could have: Advanced functionalities that would be nice to have if possible, but a bit lower prioritization than Should have.
Won't have: Extended functionality that the agent would not have, as these will not be implemented in the current software.

3.1   User Stories of features

Must have

- Implement a basic challenge/response system.

- Bank API integration (Bunq).

- A Blockchain Interface.

- Intuitive GUI.

- Able to generate a private and public key pair.

Should have

- Improved challenge/response system.

- Output the trusted public keys to the Web of Trust

Could have

- Show a list of trusted keys on the users request.

- Implement QR code scanning for exchanging public keys

- Support for 2 banks.

Won't have

- Support for more than 2 banks.

3.2   User stories of defects

Because we haven't started developing our product, we don't have user stories of defects yet.

3.3   User stories of technical improvements

Because we haven't started developing our product, we don't have user stories of technical improvements yet.

3.4   User stories of know-how acquisition

Some user stories don't revolve around adding new features, but rather performing research as to what is the best way to implement such a feature. As we develop the product, we will probably have to adapt, and add more of these user stores. Following is a list we are currently planning to do.

Compare various methods to generate public and private keys (Must have)

There are many different ways to create a public and private key pair. As they have to be generated for each user, we have to make sure the method we use is secure, while still performing in reasonable time.

Research if other banks have useable API's (Should have)

We want our app to work with as many banks as possible to be of help to the broadest range of users.

Find out the best way to store a public key in a QR code (Could have)

Typing a public key can be very tedious. If we can store the public key in a QR code, it is much easier for the user to exchange his public key with another user.

3.5   Initial release plan

Apart from the weekly scrum cycles, we plan to have 3 major releases during the development of our product. Each one will be discussed briefly.

Minimum Viable Product

The Minimum Viable Product is the first release, it contains all of the must haves specified in the Product Backlog. We will have an MVP ready for release after Sprint 4.

Improvements

Our second release, at the end of Sprint 6, all of our must haves, should haves, and most of our could haves will be integrated in our product. As we have to start the integration with other groups the sprint hereafter, we are not planning to add any more features to our product after this release.

Integration with other groups

Finally, for our third release, we will have integrated our application with those from other project groups. Our applications combined into a singular product, that provides a web of trust, the ability to enlarge this web, and the ability to improve the trust within the web.

4   Definition of done

In this section we will focus on when we consider backlog items and sprints aswell as the final product, are done.

A user story is done when it is implemented in our app, the code is tested, and approved by other developers. Tests are for example Unit tests, or manual testing. We aim for 80% branch coverage. When all tests pass, en there are no more bugs in the code, we can merge the user story into our working version.

Since a sprint is a specific time period, the sprint is done at the end of the week. If not all user stories from the sprint are implemented, they will have to be carried over to the next sprint. The work of a sprint is done when the application contains all features that were planned for that sprint. We have to run all the tests that we did for testing the features, with the addition of user tests to confirm the product works properly. We will also confirm with the client that we build the features in our sprint the way he wanted.

The final product is finished when it meets the following requirements.

- All features from our roadmap are implemented.

- All unit- and user tests have passed.

- The code is properly documented.

- We pass the final SIG test.

Only when we meet all these requirements, we can consider our product to be done.

Glossary

API Application Programming Interface. 1

blockchain is "a distributed database that maintains a continuously growing list of records, called blocks, secured from tampering and revision" (Wikipedia, 2017). 3

GUI Graphical User Interface. 2–4

IBAN International Bank Account Number. 1, 2

References

Hatton, S. (2008). Choosing the right prioritisation method. In *Software engineering, 2008. aswec 2008. 19th australian conference on* (pp. 517–526).

Wikipedia. (2017). *Blockchain — wikipedia, the free encyclopedia*. Retrieved from `https://en.wikipedia.org/w/index.php?title=Blockchain&oldid=778628009` ([Online; accessed 4-May-2017])