

# Crowdsourcing real-estate

Open market for real-estate finance

K.C. Asmoredjo  
A. Hovanesyan  
S.M. To  
C. M. Wong Loi Sing

# Crowdsourcing real-estate

Open market for real-estate finance

by

K.C. Asmoredjo  
A. Hovanesyan  
S.M. To  
C. M. Wong Loi Sing

to obtain the degree of Bachelor of Science  
at the Delft University of Technology,  
to be defended publicly on Tuesday January 31, 2017.

Student numbers: 4091760, 4322711, 4064976, 4076699  
Project duration: November 14, 2016 – January 31, 2017

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Problem description</b>	<b>2</b>
2.1	Problem definition . . . . .	2
2.2	Client description . . . . .	2
2.3	Required technologies . . . . .	3
2.3.1	Dispersy . . . . .	3
2.3.2	Blockchain . . . . .	3
2.4	Project requirements . . . . .	3
2.4.1	Must Have . . . . .	3
2.4.2	Should Have . . . . .	4
2.4.3	Could Have . . . . .	4
2.4.4	Won't Have . . . . .	4
2.5	Screens & Workflow . . . . .	5
2.6	Related initiatives . . . . .	5
2.6.1	Jungo . . . . .	5
2.6.2	Blandlord . . . . .	6
2.6.3	Greencrowd . . . . .	6
2.6.4	Fundrise . . . . .	6
2.6.5	RealtyShares . . . . .	7
2.6.6	Collin Crowdfund . . . . .	7
2.6.7	Upstart. . . . .	7
2.6.8	CrowdAboutNow. . . . .	8
2.6.9	Kapitaal op Maat. . . . .	8
2.6.10	Geldvoorelkaar. . . . .	8
2.6.11	Bouwsteen . . . . .	9
2.6.12	Gradefix . . . . .	9
2.6.13	MoneYou. . . . .	9
2.7	Development process. . . . .	9
<b>3</b>	<b>The customer journey</b>	<b>10</b>
3.1	Researching the customer journey . . . . .	10
3.2	Defining the customer journey . . . . .	10
3.2.1	The borrower's journey . . . . .	10
3.2.2	The investor's journey . . . . .	11
3.2.3	The financial institution's journey . . . . .	12
3.3	Designing the system . . . . .	12
3.4	Defining the Market API. . . . .	15
3.5	Visualising the core functionalities . . . . .	16
<b>4</b>	<b>Let's get connected</b>	<b>17</b>
4.1	Visualising the market . . . . .	17
4.2	System architecture and information flow . . . . .	17
4.2.1	Loan Request flows . . . . .	17
4.2.2	Mortgage flows. . . . .	18
4.2.3	Investment flows. . . . .	18
4.3	Integrating Dispersy . . . . .	19
4.3.1	Dispersy and Market Identity . . . . .	19
4.3.2	Dispersy messages sharing. . . . .	19

---

- 4.4 Protocol specification . . . . . 20
  - 4.4.1 API Messages . . . . . 20
  - 4.4.2 Dispersy Messages . . . . . 21
  - 4.4.3 Signed messages protocol . . . . . 22
- 4.5 Blockchain . . . . . 22
- 4.6 Document Sharing . . . . . 24

**Bibliography** . . . . . **25**

**Appendices** . . . . . **27**

- A Project description: Open market for real-estate crowdsourcing** . . . . . **28**
- B Mortgage** . . . . . **31**
- C API** . . . . . **37**
- D First version of the core screens** . . . . . **46**



# Introduction

The mortgage market is a multibillion market that has seen some serious lows in the last couple of years. The high amount of borrowers defaulting their mortgage played a big role in the 2007-2008 financial crisis [40]. That financial crisis had a big influence on the world economy.

Taking a mortgage from Dutch financial institutions can be done as follows. First, the borrower needs to decide what the possibilities are, e.g. what are the total costs and how big can the loan be? When the financial situation of the borrower has been verified, a house can be chosen that fits the borrower's needs. When an agreement has been met with the seller of the house, a mortgage can be settled with a mortgage lender. Together with the provider it will be possible to choose for a mortgage type that fits the borrower's financial situation.

In the Netherlands, the current maximum mortgage limit is 102%. Each year it is being lowered with 1% till 2018, when the maximum mortgage limit will be 100% [27]. The CEO of the Dutch National Bank [11], Klaas Knot, wishes to see the maximum mortgage limit be reduced to 90% by 2028 [39]. While the maximum amount of the mortgage is a legal limit, financial institutions will usually only provide a fraction of the property value amount. How much the financial institutions will provide exactly depends on the Loan-to-Value ratio [22], which differs case to case. Instead it is expected that the borrower provides a downpayment of the remaining value. The large downpayment means that people that are looking to buy a house will have to save up before they can buy a house.

To enable buyers to get the capital they need but do not have, crowdfunding is a solution. Such crowdfunding platforms already exist and they are becoming more popular. Some give the opportunity to crowd-fund mortgages while others are more general and provide a place for entrepreneurs to find investments for their projects. These projects range from opening a new shop to projects like installation of personal solar panels. What these platforms all have in common is that they are centralized, which means that there is a third party that connects the investors with the investees. The crowdfunding services they provide are not for free, which results in unneeded costs for the users of the platform. A way to eliminate the unneeded costs is to make a decentralized platform, where no middleman is needed.

# 2

## Problem description

In this chapter we define the problem, give a description of the client, and analyze related initiatives. Further it contains the project requirements and the description of the screens we have designed.

### 2.1. Problem definition

It is currently very difficult for an average citizen to get a mortgage. Only borrowers with minimal risk of defaulting are given a mortgage. As the value of houses keep rising, it becomes increasingly difficult for people to receive a mortgage. Additionally, because of the mortgages not covering the total cost of the estate, the only possibility is to make a large down payment. For a lot of people, and especially for young adults that recently joined the working force, down payment is not an option. At the same time, interest on saving accounts is so low that it does not covers inflation. Investing could bring a solution to savings losing value over time.

There is a need for a system that connects these parties together. The parties consist of the **borrower**, the **investor** and the **financial institution**. These three stakeholders all have their own incentive to work with each other. The borrower needs a bank to settle a mortgage and investors to fund the initial down payment. In return the public investors and the bank can receive interest on their respective investments. To solve the mentioned problems, we focus on creating a platform where mortgages can be partially crowdfunded.

Crowdfunding platforms are getting increasingly popular to create funds for projects in which banks are not investing. Platforms on which it is possible to get your mortgage crowdfunded already exist, yet on all of these platforms there is a central party that connects the investors with the borrowers. The central party is not only unneeded, but also brings unnecessary costs and risks. Bringing down the middleman will also bring down the whole system.

To avoid that the whole system collapses when one of the key parties drops out, the system needs to be decentralized. During the project, we will strive to create a platform where people can find mortgages that are funded by both banks and investors. A person will be able to apply for a mortgage where the bank can fund a fraction of the loan (e.g. 70% of the total loan). When the mortgage has been accepted the loan request will be transferred to an open market place where investors can crowdfund the remaining part of the mortgage. The interest rate on these investments will be negotiated between the borrower and the investor.

The aim of this project is to create a scalable decentralized market, where it is possible to crowdfund a mortgage. The system will not be controlled by a single party, but mostly by supply and demand.

### 2.2. Client description

The client for this project is ABN AMRO [3]. ABN AMRO is collaborating with the TU Delft Blockchain Lab [6], which is part of the Tribler team [32], on the development of blockchain [5] applications. They aim to create some complex blockchain applications that are suitable for use in practice, within half a year [34]. With the collaboration, ABN AMRO wants to improve their knowledge on blockchain technology, especially on the practical application of distributed digital ledgers.

TU Delft has been researching blockchain since 2007. The research at TU Delft focuses on developing next-generation blockchain technology that is trustworthy and scalable. Their scalable blockchain is based on tamper-proof secure timelines and graph-based data structure. This project will be built on previous work that has been done at the Blockchain Lab [33]. All of their work is shared with the open source community.

## 2.3. Required technologies

Being a decentral application, there can be no central server. All communication must be shared between actors in the system, with no central point of trust or authority. This can be achieved by creating a peer-to-peer network for the decentral real-estate market application. Originally the assignment was to extend the existing decentral market [37] to include crowdfunded mortgages. This market community was built upon the Distributed Permission System (Dispersy) [10] for peer-to-peer networking. Thus using Dispersy will be required.

Furthermore, this being a project for ABN AMRO in collaboration with the TU Delft Blockchain Lab for development of blockchain applications, it is expected that the market utilize blockchain technology in some way.

### 2.3.1. Dispersy

Dispersy is a fully decentralized platform designed to simplify the creation of distributed communities. It uses elliptic curve cryptography [36] in their simple identity and messaging system to securely and anonymously identify different nodes. It can run on systems consisting of a large number of nodes, without having the need for any server infrastructure. Each node is equally important and they run the same algorithm to perform the same tasks, which increases the robustness of the system. Data is forwarded between nodes and will eventually reach all nodes.

### 2.3.2. Blockchain

A blockchain is a distributed database that holds a steadily growing list of data called *blocks*. Each block contains, among other things, a timestamp and a link to the previous block. Within the system, data is being exchanged and authentication between the nodes takes place. The data can not be manipulated or falsified, due to the distributed system. Using a blockchain it is not necessary that a third party ensures the reliability of a transaction.

## 2.4. Project requirements

To prioritize the requirements, we used the MoSCoW method [24]. The requirements are subject to change during the development phase.

### 2.4.1. Must Have

The Must Have provides the Minimum Usable Subset (MUST) of requirements that the project guarantees to deliver. These are the core functionalities and without these, the product cannot be delivered. These requirements have the highest priority.

- Borrower can place loan requests.
- Borrower can upload documents needed to apply for a mortgage.
- Borrower can see offers they get from investors and financial institutions.
- Borrower can accept offers they get from investors and financial institutions.
- Investor can see which campaigns are available.
- Investor can place an offer on a campaign.
- Financial institution can create a quote for a mortgage.
- Financial institution can see their pending loan requests.
- Financial institution can review pending loan requests.
- Financial institution can accept pending loan requests.
- Python 2 to be able to interface with Tribler and Dispersy.
- PyQt for the GUI.
- Tests and coverage.
- Full transparency. Everything open-source.

### 2.4.2. Should Have

The Should Have requirements are important but not vital. The product is still viable without these functionalities. It has the highest priority after the Must Have requirements.

- Borrower can reject offers they get from investors and financial institutions.
- Investor can see which campaign they have currently invested in.
- Financial institution can see which mortgages they currently have provided.
- Financial institution can reject pending loan requests.
- Scalable.
- Blockchain technology.

### 2.4.3. Could Have

The Could Have requirements are wanted or desirable, but less important than the Should Have requirements.

- Investor can resell their investment.
- Investor can invest passively.
- Financial institution can determine the maximum interest rate that a borrower is allowed to pay an investor for the loan.
- Financial institution can recommend an interest rate that the borrower can pay to an investor for the loan.
- Secure storage and transfer of information.
- Encrypt and decrypt user documents.

### 2.4.4. Won't Have

The Won't Have requirements will not be implemented in the final solution, because they are out of scope for this project.

- Borrower can see how much has already been paid off.
- Investor can see how much has already been paid off.
- Financial institution can see how much has already been paid off.
- Regulator can see the total financial health of all the active loans.
- Transactions can be done through the system.
- Smart contracts to ensure binding agreements between stakeholders.
- The system can do a risk assessment.

Smart contracts are out of scope for this project and will be researched and implemented by a PhD student after we have finished this project. Besides smart contracts, transactions through the system are out of scope for this project too. Because transactions can not be done through the system, borrowers, investors and financial institutions will not be able to see how much has already been paid off. Therefore the regulator can not see the total financial health of all the active loans. We have decided that the system will not be able to do a risk assessment, as this would be too complex and will not be achievable in the duration of this project.



## 2.5. Screens & Workflow

Based on the functional requirements, a list of pages the program should have to achieve these goals has been compiled. The screen can be either for borrowers [B], investors [I] or financial institutions [F].

### 1. Profile [BIF]

The profile screen differs depending on the role of the person that is logged in. It is used to save and update personal information. Investors only need to enter their name, email address and IBAN. Borrowers need to enter more personal information and they can upload personal documents such as payslips and proof of identity.

### 2. Portfolio [BIF]

The portfolio screen differs depending on the role of the person that is logged in. Borrowers can see, accept and reject the offers they got from banks and investors. Investors can see the loans they invested in, their pending investments, and they are able to resell their loans to other investors. When they are reselling a loan, they can accept or reject the offers for that loan. Financial institutions can see which mortgages they currently have.

### 3. Place Loan Request [B]

Borrowers can place a loan request, which will then be sent to all financial institutions of their choice. This will include information on the house, its price, and how much money they want to borrow. There will also be a box to write a small description, so an investor can learn more about who they are investing in.

### 4. Pending Loan Request [F]

Financial institutions can see all pending loan requests. They can review each of the requests and accept or reject them. When they accept a loan request, they have to enter the details about the mortgage, such as interest and duration.

### 5. Place Loan Offer [I]

This screen allows investors to invest passively. They can enter the amount they want to invest, the interest they want, and the duration of the loan.

### 6. Open Market [BIF]

On the open market all investing opportunities are presented to all roles, but investment can only be made by investors. Key things to show are the rating of the mortgage, the duration of the loan, risk and type (new or aftermarket). Search and filter functions will be included as well.

### 7. Campaign Bids[BIF]

The Campaign Bids screen can be reached from the open market by selecting a campaign, and shows the current bids that investors have made in the campaign. Additionally, from here it is possible for investors to place a bid of their own by entering the amount they want to invest, the duration of the loan and desired interest rate.

## 2.6. Related initiatives

To get some more insight into the current crowdfunding market it is necessary to analyze initiatives related to crowdfunding, crowdfunded mortgages and crowdfunded loans. A quick Google search has returned numerous examples, of which 13 will be analyzed in this section.

### 2.6.1. Jungo

Jungo [19] is a mortgage bank, with the ability to crowdfund your mortgage. All clients whom are accepted by the bank will guaranteed get a mortgage. But Jungo differs from normal mortgage banks by making it possible for 3rd parties and institutional investors to provide a part of the mortgage in return for an attractive interest.

At the moment of this writing it is not yet clear what the application process is for a mortgage via Jungo as they have not yet launched their service. All borrowers will undergo a risk assessment based on the 'Code of Conduct for Mortgage Loans' [1].

Investors can choose which project they want to finance, for an amount between €250 and €80.000. The provisional interest is between 3% and 3.5%. They pay the money via an iDEAL transfer to the person requesting funding, and will be paid back monthly for up to 8 years. The length of the investment varies per Jungo campaign.

Investors are protected from borrowers who are not able to live up to their obligation by a fund controlled by Jungo to continue paying investors when a borrower misses a payment. This fund is limited however.

Thus Jungo can be seen as a mortgage bank that also functions as intermediary for individuals and institutional investors who wish to invest in a mortgage. Finally they also provide a guarantee (up to a certain point) that your investment will be returned.

### **2.6.2. Blandlord**

Blandlord [4] is a platform for crowd ownership of homes. The idea behind the platform is that home owners can put their house for sale via Blandlord, even if the home is inhabited (the new landlord of the renters will become Blandlord). These houses can then be bought by a number of investors who can buy a percentage of the house. The rent income is then shared among investors depending on the percentage that they own. Thus Blandlord is not only a platform for selling the houses, it also functions as a letting agent.

All homes must be 100% owned by the owner before being placed on Blandlord. To minimize the risk that the homes stay empty due to lack of renters, only houses in areas with a healthy house market are accepted, as judged by Blandlord. Furthermore, an extensive audit of the condition of the house is required as a prerequisite of being able to list the house on Blandlord, to prevent the investors being faced by unforeseen costs.

Investors can buy shares in a house, with the minimum investment amount being €100. The value of the shares are directly correlated to the value of the house. They are paid their share of the rent on a monthly basis, based on the expected rental income. The costs of using the service is a 1% transaction fee on all transactions (buying/selling shares).

### **2.6.3. Greencrowd**

The focus of Greencrowd [17] is to provide a crowdfunding platform for sustainable energy projects such as installation of solar panels, wind power, et cetera. These projects will in turn provide income by selling the power or service, or for instance lower the energy costs of a building that has installed solar panels.

Organisations can present a plan for a sustainable energy project, which will then be analyzed by Greencrowd to make sure the plan is realistic. They will also analyze the financial risks and assign a grade to each project, ranging from A to G. The interest an investor can expect is tied to the grade. The riskier the project, the higher the return. A project has a certain amount of time to get financed, if the finance goal is not reached before the deadline, all investors will be returned their money.

Investors can invest in all running projects, with the amount varying per project. The running time of the loan also varies per project. The only constant is that the interest is always tied to the grade the project received. Payments also vary per project, which can be monthly or quarterly, and they can include special provisions for paying off a loan earlier (the investor then receives a bonus from the investee).

### **2.6.4. Fundrise**

Fundrise [12] is an online platform where individuals, accredited or non-accredited, are able to directly invest in commercial real estate projects from real estate companies, without the middleman. The main difference between the Fundrise eREITs (electronic Real Estate Investment Trust) and other REITs, is that Fundrise is more transparent and has roughly 90% lower fees. Fundrise enables investors to choose in which eREIT they want to put capital in.

To submit a project for a loan, the companies need to indicate the amount to raise and the project type, and they have to upload relevant documents needed for the application. First the information will be screened to ensure investment is consistent with Fundrise standards. Then follows the in-depth analysis, which includes an on-site visit by a member of the Fundrise team. When the project is accepted, it will receive a Fundrise rating, a letter rating ranging from A to E so that investors can easily compare different investments.

The minimum investment is \$1.000 and the maximum may not exceed 10% of the investor's gross annual income or net worth. The fund transfer is done via bank account. When investors get their return is specific to each listing. When available, a distribution schedule can be found in the offering documents for that investment. Generally investors can expect to receive distributions quarterly. Asset management fees are between 1% and 1.5%.

In the event that a company is not able to pay back the loan, the eREIT will bear the risk. The eREIT will pay the investors to the extent of any deficiency between the value of the collateral and the amount of the loan.

### **2.6.5. RealtyShares**

RealtyShares [26] is a real estate crowdfunding platform that lets accredited investors choose what they invest in. Investors have the choice to invest in one specific property or in a group of properties. RealtyShares offers commercial and residential real estate projects.

To apply for financing, relevant information and documents should be provided by the borrower. RealtyShares then reviews whether the borrower prequalify based on their track record, financial strength and expertise. After the pre qualification, borrowers will be thoroughly reviewed based on their investment strategy, financials, legal standing, and property condition/location.

Investors can start investing from \$5,000. The funds transfer is via bank account. Depending on the type of investment, investors get their return either quarterly or monthly. Investor fees have a maximum of 2% of the invested amount. RealtyShares offers investors to loan their money supported by a collateral, a specific real estate property or pool of properties.

### **2.6.6. Collin Crowdfund**

Collin Crowdfund [8] offers linear loans through crowdfunding to established businesses and startups. Innovatively Collin returns to the basics of banking, where the investor decides what they invest in. Collin also cooperates with banks to demonstrate what is realistic and to give the business the opportunity they deserve. That combination ensures that the investor makes a good return at a reasonable risk and that the business can realize their ambitions.

To apply for a loan the business has to indicate the desired loan amount and the purpose for the loan, and they have to upload relevant attachments for their loan application. Collin then provides the business with a Crowdfund Coach to assist with filling out the Collin Credit Score model, which is a risk assessment that tests the company in terms of solvency, profitability and liquidity. The Collin Credit Score produces a D&B rating [2] and the boundaries of the interest rate the business is allowed to set. Collin either approves or rejects the loan application using all of that information.

Investors can choose which projects they want to finance. Private investors can invest for an amount between €500 and €80,000 and corporate investors can invest from €500 with no upper bound. They pay the investment via an iDEAL transfer to the business requesting funding and will be paid back monthly for the duration of the project, which can be anywhere between 6 months and 10 years.

To protect investors from businesses that are not able to live up to their obligation, Collin continues paying investors for one month. When the business is not able to pay the investors back after that month, there will be no payment of interest to investors.

Collin asks for a small fee from both the business and the investors, which makes it possible for Collin to take care of the payments for both sides. For every investment a management fee is calculated annually at a rate of 0.85%. The business is asked for a one-time fee of 1.9% for a successful loan and a monthly fee of 0.05%.

### **2.6.7. Upstart**

Upstart [35] is a platform that allows people to obtain loans with a fixed rate. Upstart does not only look at credit scores, because they believe people are more than just credit scores. They use a model that considers the school the borrower has attended, their field of study, their academic performance, and employment history in addition to their credit history. That allows Upstart to offer people the loans they have earned.

To apply for a loan at Upstart, borrowers have to fill in an online application which will include information about their academic credentials, work experience and what they plan on doing with the loan proceeds. Upstart also verifies the borrower's personal and credit information as part of the application process.

Investors cannot choose in which project they invest, because loans are not visible publicly on Upstart. Investors can invest for a minimum investment amount of \$100. All loans are fixed-rate loans with a 36 or 60-month repayment period. Investors will get an investor account on Upstart, to which they can transfer money to invest or to transfer their returns to their own bank account.

Upstart determines the interest rate using the borrower's education, credentials, work experience, and credit history. This means that the interest rates are different for each loan. Once the interest is determined,

all loans have a fixed-rate interest. The loans are unsecured obligations of the borrower and are not supported by any collateral.

As compensation for the marketing and ongoing servicing of loans, Upstart receives servicing fees that are paid from the origination fees that are collected from borrowers. Borrowers pay a one time origination fee that is between 1% and 6% of the target loan amount. Investors pay an annual servicing fee of 0.5%.

### **2.6.8. CrowdAboutNow**

CrowdAboutNow [9] is a crowdfunding platform that makes it possible for entrepreneurs to start a crowdfunding campaign. They will support the entrepreneurs whenever they need it. CrowdAboutNow makes it possible to invest safely, both legally and technically.

To apply for a loan, entrepreneurs have to take a test to determine if their plans are suitable for crowdfunding. If their plan is suitable, an account manager will get in touch with the entrepreneur to discuss their plans.

Investors can choose the projects they want to invest in themselves. Investors can invest for an amount up to €80.000. They pay the investment via an iDEAL transfer, for which a fee between €0.70 and €0.80 is asked. It is not clear in what period of time and at what interest rate the investors will get their investments back.

To minimize the risk for investors CrowdAboutNow verifies if the entrepreneur is willing to use its own network to invest its venture capital. CrowdAboutNow verifies the identity of the entrepreneur, the registration of the company, and its bank account. They also meet with the entrepreneur in person. If the entrepreneur has been registered for less than 3 years at the Chamber of Commerce [21], a partner that understands the industry will act as the nominating party. Should there not be enough investors to reach the loan target, all investors will get their money back.

CrowdAboutNow advertise entrepreneurs' crowdfunding campaigns, they give legal advice, and take care of the financial transactions. They do this for a one-time entry fee of 1% of the loan target and a 2% annual fee when the campaign is successfully funded.

### **2.6.9. Kapitaal op Maat**

Kapitaal op Maat (KoM) [20] is a platform that makes it possible for different projects to get crowdfunded. Possible project funding sum can range from €25.000 to €500.000. Individual users can invest any amount between €100 and €50.000. Corporate investors do not have a hard ceiling, but both types of investors cannot invest beyond the amount that is needed to get the project fully funded.

To apply for a loan the company or entrepreneur needs to present the website with a business plan of the venture. After that a fee of €395 is paid to KoM to examine the presented offer and determine the risk of the venture. Additionally, if the venture is accepted, another fee of €350 must be paid to publish the venture on the website. The borrower will receive guidance with the right presentation of the offer. The calculated risk of the venture, which consists of a grade from A+ to C-, determines the interest. This will range from 5.5% for the lowest risk ventures (A+) and 9.5% for the highest risk ventures (C-).

When an investment is made, an initial fee of 0.9% of the investment is paid. All the investments are sent to the escrow account of KoM and stored until the the funding goal is reached. If this goal is reached, a fee of 4% must be paid by the borrower to receive the funding. When the fees are paid and the transactions are made, the borrower will continue to pay the escrow a fixed rate every month until the investment and interest are completely paid off. After that, the escrow distributes the amount accordingly among the investors.

### **2.6.10. Geldvoorelkaar**

Geldvoorelkaar [13] is a crowdfunding platform where Dutch corporations and individuals can invest in national, small to middle sized ventures.

The platform provides different types of loans which consist of: Fixed-rate, Interest-only and Convertible debt loans. The duration of the loan can range from 6 to 120 months. This can be as little as €1.000. There is no limit to how much an entrepreneur can ask, but no funding will be received until the loan goal is reached.

The fees for a funding request are €125 for private investors and between €350 and €500 for corporate investors. Additionally, when the loan goal is reached, the borrower pays a small fraction of the loan goal as success fee each year for the duration of the loan. This fee can range from 0.5% to 7.9% of the loan goal.

Geldvoorelkaar does background checks on everyone asking for a loan. Before a request can be made, the borrower has to present a business plan and a financial report. This is used to determine risk of the venture, which in turn will be used to determine interest of the loan. The higher the risk, the higher the interest. This

is usually somewhere between 4% and 10% interest. After the transactions are made, the investors receive monthly payments by the borrower until the loan and interest are completely paid off.

### **2.6.11. Bouwsteen**

Bouwsteen [7] is a new initiative that is going to be available in Q1 of 2017. They want to provide a solution targeted at groups that are not served adequately in the market yet, namely starters (employees and self-employed) and seniors. Their goal is to create a financing product for the private housing market that allows for a reduction in the monthly payments, providing flexibility to the homeowner. Since Bouwsteen have not launched their service yet, it is not clear in what way the mortgages they will offer will work and how they are planning on reducing the monthly payments.

### **2.6.12. Gradefix**

Gradefix [16] is a new initiative by ABN AMRO to automate credit checks. It is completely different than the previous initiatives described in this section as it has nothing to do with crowdfunding, but credit checks are an integral part of any mortgage application. Gradefix calculates this score on the basis of the requester's bank transactions. By calculating the incoming and outgoing sums, and how much money is left at the end of each month, a credit rating is generated.

### **2.6.13. MoneYou**

MoneYou [23] is a bank that gives borrowers a lot of freedom with their mortgage. Unlike other banks, MoneYou does not give any advice in any form. It is up to the applier to decide what is best for them and their situation. The applier decides the specifics of the mortgages such as mortgage type, loan duration and loan amount.

What makes MoneYou special compared to other banks is that they facilitate the mortgage and all of the steps to get a mortgage online. In other words, it is possible for people to apply for a mortgage and go through most of the steps online. Before an application can be made the applier needs to do a quiz on the MoneYou website. The quiz consists of multiple choice questions that should show that the applier knows what they are doing and has the financial knowledge and assets to take out the mortgage. The quiz and other submitted documents such as payslips and other financial documents are used by MoneYou to make a final verdict on giving out the mortgage or not. After the mortgage has been set, the definitive purchasing of the real-estate can be made with the help of a notary.

## **2.7. Development process**

During the project we used the Scrum framework [28]. We worked in sprints of 2 weeks, and we focused on finishing a new functionality every sprint. Each chapter of this thesis reflects one sprint.

We held daily meetings within our group, we held meetings with our coach every week to discuss our process and progress, and we held meetings with our client every few weeks to discuss our progress. The meetings made sure that we were able to stick to the goals of the client and to incorporate feedback that they provided. We developed the software using GitHub [14] for revision control. During the the development of our software we made use of Travis CI [31] and Jenkins [18] for continuous integration. Our project code can be found at <https://github.com/Jumba/mockchain-market>.

# 3

## The customer journey

After centering our first two-week sprint around the orientation of the problem, we started working towards a demo for our client. We have also designed the system, defined an API and started working on the core functionalities in the GUI.

The problem has three equally important stakeholders, as the system can not be functional with any of these stakeholders not present. These stakeholders are the borrowers, investors and financial institutions. In the following sections, the journey of each of the stakeholders will be discussed.

### 3.1. Researching the customer journey

Throughout the first sprint of the project, we concentrated on understanding the problem and related initiatives. Through analyzing the different initiatives, we came to understand more about crowdfunding real-estate.

Because our problem concerns people who need a mortgage, we had to understand how mortgages work too. The two most common mortgages are Linear and Fixed-Rate [25], which are available for starters. Bart Gout, our real-estate/mortgage expert, provided us a document that listed the required documents needed for applying a mortgage and a scheme showing the process of getting a mortgage. These documents can be found in appendix B.

From the related initiatives, which were described in section 2.6, we found out that there are existing crowdfunding platforms for real-estate. There is even a bank that facilitates the mortgage and regulates all the steps to get a mortgage entirely online [23]. A MoneYou representative explained that the required identity check can be delayed until the end when the borrower meets with the notary. Only the definitive purchase will have to be made with a notary. Thus the marketplace does not need to supply a way to verify the identity of a borrower.

### 3.2. Defining the customer journey

With the required information for the application process of a mortgage in mind, and thinking about what the financial institutions need in order to offer a mortgage, and considering what the possible options are for potential investors, we started defining the customer journey.

#### 3.2.1. The borrower's journey

The borrower's journey can be seen in figure 3.1 and starts when the person opens the market software. At this stage are two cases, either the person has used the software before or they have not. When they have used the software before, they can supply their private key to identify themselves. When they have not used the software before, a key pair has to be generated for the user. After receiving their key, this person is not yet a borrower, as this is a general step for both borrowers and investors.

After logging in they will be forwarded to their profile. On this profile page they can add or append all required personal information. The first time they fill in the profile they choose their role, which can be either borrower or investor. In this section the chosen role will be 'borrower'. Once the required fields are filled in and all the needed documents are uploaded, a prospective borrower can go on to apply for a mortgage. If

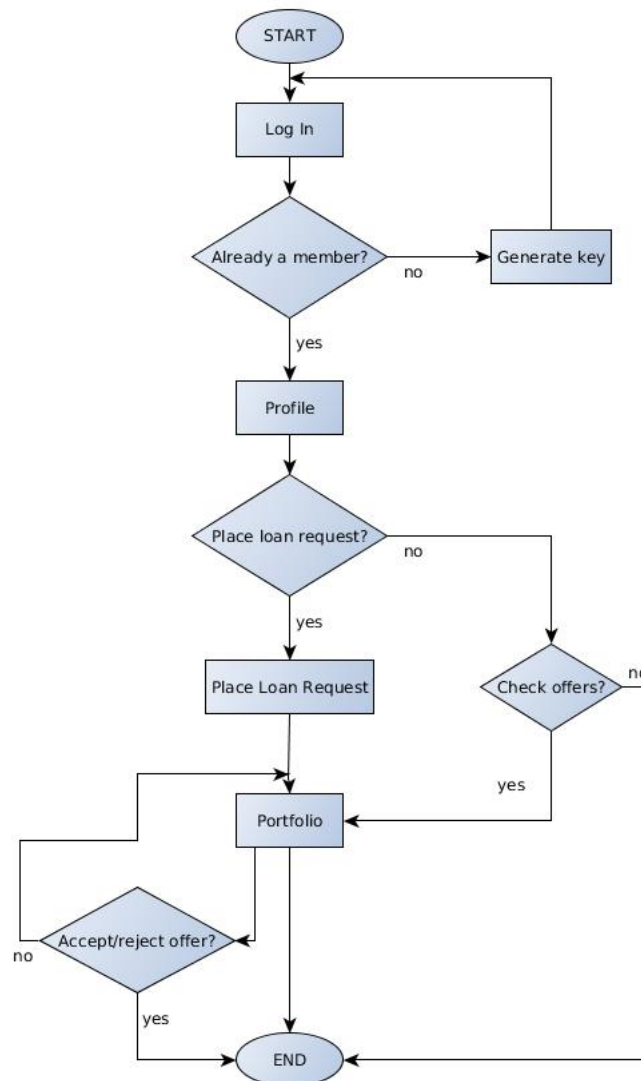


Figure 3.1: Borrower's flowchart

they have already applied for a mortgage, they can check to see if any offers have been received from banks or investors in their portfolio.

Applying for a mortgage is a fairly straightforward process as all required documents will have been uploaded to the profile. The user will have to supply mortgage-specific information, such as house information, the real estate broker, et cetera. They can also choose which banks to send the request to. Once sent, the only course of action is to wait for offers from the banks.

Once a mortgage offer is received, it will be shown to the borrower in their portfolio. The borrower can then elect to accept the offer, or reject the offer. Once a mortgage offer is accepted the crowd funding campaign will start. At this point the mortgage offer is placed on the open market where investors can submit investment bids. As with the mortgage offer from banks, offers from investors will be shown in the portfolio.

From now on the market software will take care of all the work until the campaign is either funded, or the time limit has passed.

### 3.2.2. The investor's journey

The investor's journey can be seen in figure 3.2 and starts, like the borrower's journey, when the person opens the market software. They too, have to identify themselves using their private key if they have used the software before, or generate a new key pair if they are new users.

Once they have identified themselves, they are forwarded to their profile. There they can edit their per-

sonal information, such as contact information and IBAN. If it is their first time filling in the profile, they have to choose their role, which is either borrower or investor. In this section the chosen role will be 'investor'. After the required information is filled in, they are able to see their portfolio, search for investment opportunities, or choose to invest passively.

The portfolio page lists all of the investor's current investments, pending loan offers, and bids on their investment that they want to resell. They can monitor their investments, accept or reject investment opportunities and bids, and choose to resell investments.

If the investor chooses to actively search for investment opportunities, they can do so at the open market. When an investment opportunity has been selected, they can see the current investment bids on that particular opportunity and place a bid of their own. Their investment bids and the status of the bids will be shown in their portfolio.

The investor can also choose to invest passively. All they have to do is fill in the amount they want to invest, the approximate interest rate they want to receive for it, and the duration of the investment. When the loan offer is placed, the market software will automatically search for investment opportunities that matches the loan offer. The moment an investment opportunity has been found and is accepted by the borrower, it will be shown in their portfolio.

Ongoing investment opportunities can be placed on the open market, if the investor wants to resell their investment. They have to fill in the amount and the interest rate they want for it. Once other investors have placed their bids, the bids will be shown in the investor's portfolio for them to accept or reject.

### 3.2.3. The financial institution's journey

The financial institution's journey can be seen in figure 3.3 and starts, like the borrower's and the investor's journey, when they open the market software. The financial institution has to identify themselves by supplying their private key. Unlike the borrowers' and investors' keys, their public key is already saved in the software to ensure that people can not pretend that they are a financial institution.

Once the financial institution has identified themselves, they will be able to see their portfolio. In their portfolio they can see the information on all of the mortgages that they currently have running.

From their portfolio they will be able to go to another page, on which their pending loan requests are displayed. On this page there is a list of all the loan requests from people who want to take a mortgage have submitted.

The financial institution is also able to view every one of those loan requests individually. On the individual loan requests page they will be able to view personal information about the borrower, including the documents that are needed to apply for a mortgage, and mortgage specific information, such as the house the borrower wants to buy and the amount of money they want to borrow. On this page they have the option to accept or reject the application. When they want to accept an application, they will have to fill in some details on the mortgage they want to give out. Once the application has been accepted, it will be sent to the borrower who can then decide if they want to accept or reject their offer.

Financial institutions also have the option to explore the open market. They will be able to see all current running campaigns and the bids that investors have placed on them, but they will not be able to place bids themselves.

## 3.3. Designing the system

In this section the object-oriented analysis will be documented. This analysis will use the problem description, as described in chapter 2, and the customer journey as the source to determine which objects will need to be represented in the system, how they will behave, and how they will interact with each other.

The first object identified is the **User**. This 'User' can be either a 'borrower', 'investor' or 'financial institution' which are immediately defined as **Roles** within the system. To simplify the system a user can only have one role. Depending on the 'role', certain information will be required from a user. For instance, it is required to know the current address of a borrower, since the financial institution needs to know where to send mail. But this information is not needed for the investor. Based on this, a **Profile** object has been created which includes all personal information of a user. To accommodate the extra information required for a borrower a **BorrowersProfile** has been created, which extends the 'Profile'. For a mortgage request, an array of documents is required. These will be stored in a **Document** object.

When a borrower wants to request a mortgage, a **House** object is created with details of the house and its price, which will be sent to the financial institution along with a **LoanRequest**. In this 'LoanRequest' the



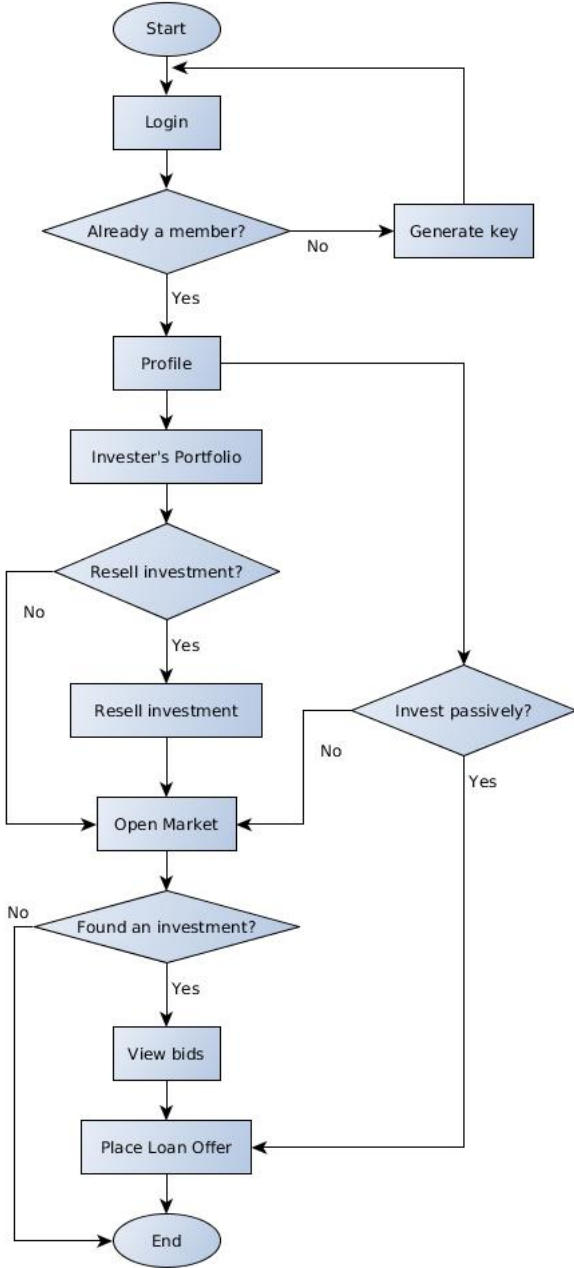


Figure 3.2: Investor's flowchart

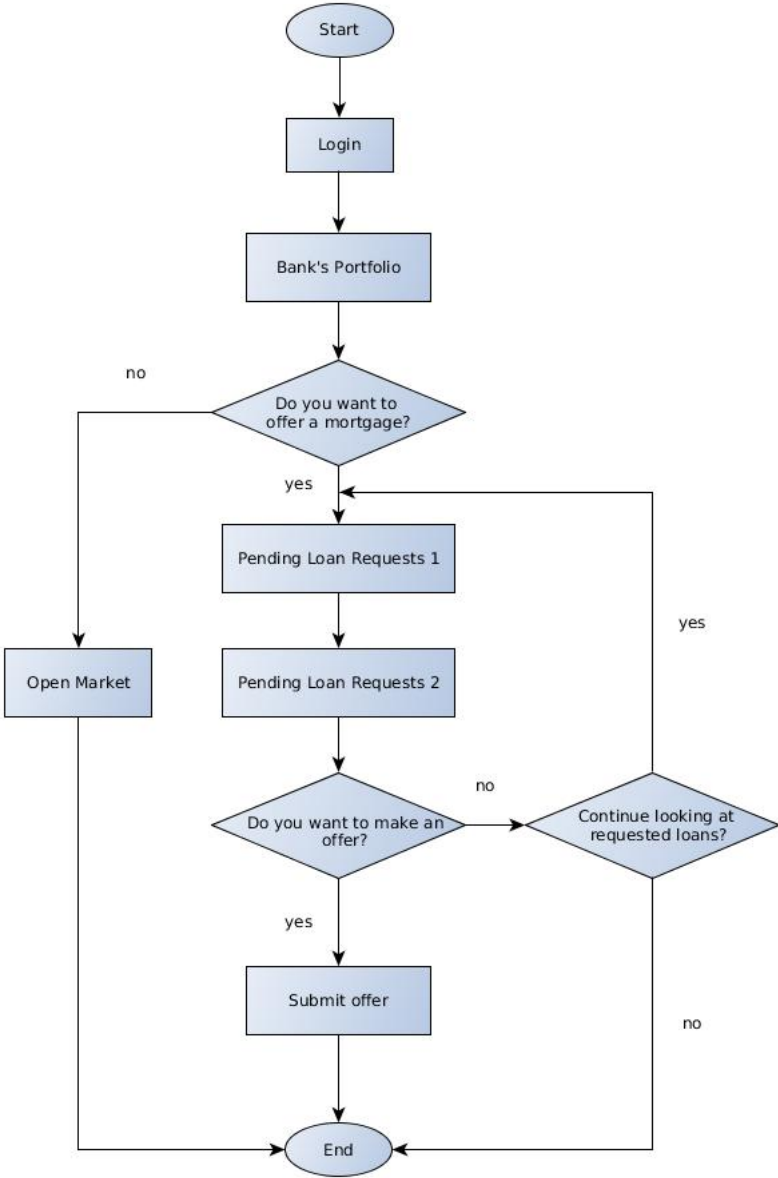


Figure 3.3: Financial institution's flowchart

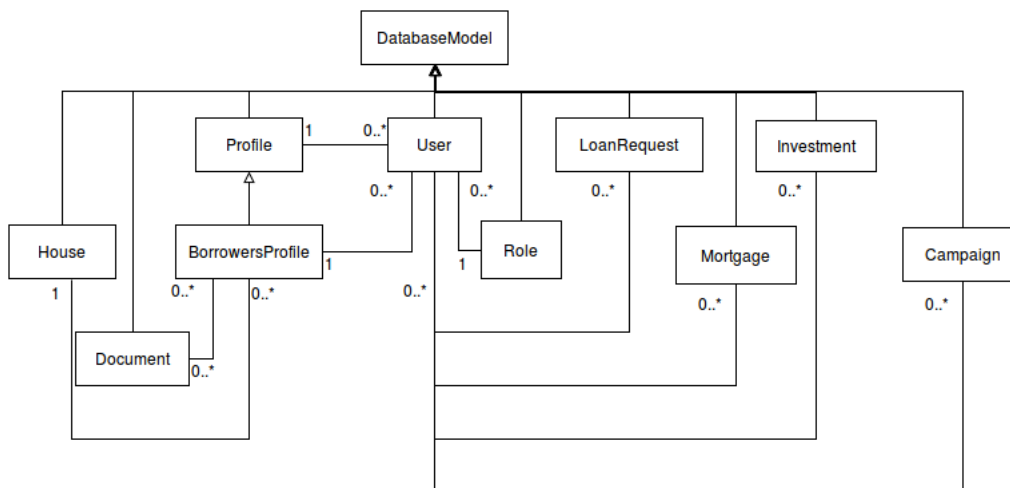


Figure 3.4: Class diagram

borrower can specify which type of mortgage, amount wanted, et cetera. As soon as a financial institution accepts the offer, this request is turned into a **Mortgage** which can be placed on the open market as part of a **Campaign** to crowdfund the remaining part of the mortgage. Finally, investors can start investing in mortgages by submitting **Investment** offers.

Since all objects will have to be encoded to be sent over the network, a **DatabaseModel** has been created from which all the other objects descent. This 'DatabaseModel' implements functions to encode and decode the objects and generating unique IDs. The relations between these objects can be seen in figure 3.4.

### 3.4. Defining the Market API

To ensure that 3rd-party programs can seamlessly access the decentralized mortgage market an API has been designed. The goal of this API is to provide all functions required to be able to complete the customer journeys.

Currently the API is *Python* based, with most 'POST' functions having the signature *function(user, payload)*, where payload is a dictionary. It has been decided to use payloads for each API call to make the addition of an HTTP layer implementing a RESTful API simpler. In theory the JSON calls can be directly converted to a dictionary, which means direct conversion from JSON to the payload will be possible.

Based on the customer journey analysis, the following API functions have been created:

- create\_user()
- create\_profile(user, payload)
- create\_loan\_request(user, payload)
- create\_campaign(user, payload)
- place\_loan\_offer(user, payload)
- login\_user(private\_key)
- load\_profile(user)
- load\_investments(user)
- load\_open\_market()
- load\_borrowers\_loans(user)
- load\_borrowers\_offers(user)
- load\_all\_loan\_requests(user)
- load\_single\_loan\_request(payload)
- load\_bids(payload)
- load\_mortgages(user)
- accept\_mortgage\_offer(user, payload)
- accept\_investment\_offer(user, payload)
- accept\_loan\_request(user, payload)
- reject\_mortgage\_offer(user, payload)
- reject\_investment\_offer(user, payload)
- reject\_loan\_request(user, payload)
- get\_role(user)
- resell\_investment()

The full API documentation has been included in appendix C.

### **3.5. Visualising the core functionalities**

Defining the journeys and the flows of the customers gives a good understanding of what functionality the user interface has to provide. The program's core functionality can be summarized by the following actions:

- Placing a loan request (done by a borrower as a request to receive a mortgage)
- Reviewing loan requests (done by financial institutions to assess a borrower's financial situation and to determine if they are eligible for a mortgage)
- Viewing campaigns (a place where all of the campaigns that can be crowdfunded are displayed)
- Placing bids on campaigns (a place where bids on a campaign can be viewed and investors can bid on the campaign)
- Accepting bids on campaigns (the campaign owner can view bids on their campaign and choose the ones that they find profitable)

From these core functionalities, the first version of the GUI was made. The rest of the screens exist to make the whole user experience go smoother and the core functionalities more accessible.

# 4

## Let's get connected

In this sprint we focused on finishing the GUI, transferring documents, and integrating Dispersy and the blockchain into the market.

### 4.1. Visualising the market

In section 3.5 we described the core screens that we implemented. In this section we will discuss the rest of the screens. All of the screens, including the core screens, can be found in appendix D.

To make sure that financial institutions can access personal information about potential borrowers, a profile screen for borrowers has been created. In that screen they can fill in their personal information and also upload documents that are needed to apply for a mortgage. Investors can also see the borrower's name and IBAN when their investment has been accepted.

The borrower needs to know the name and bank account of investors that have made an investment on their campaign, to be able to pay them. To make sure that they can see that information, the profile screen has also been created for investors.

A portfolio screen has been made for borrowers to view their current loans and loan offers. It makes sure that they can keep track of their campaign and the loans that are tied to that campaign. In the same screen they can see loan offers, and also accept or reject them.

Investors should be able to see which investments they currently have running. To make sure they can see their investments, a portfolio screen has been made for investors.

### 4.2. System architecture and information flow

There are three big stages to the software, which are sending, accepting or rejecting a loan request, sending, accepting or rejecting a mortgage offer, and sending, accepting or rejecting an investment offer. For each of those stages we have designed protocols. The stages each apply to a model, which are the **LoanRequest**, **Mortgage**, and the **Investment** models respectively. The model specification can be found in section 3.3.

#### 4.2.1. Loan Request flows

When a borrower sends a loan request to a financial institution, the financial institution can respond by either rejecting the loan request or offering a mortgage to the borrower. The loan request flows can be found in figure 4.1.

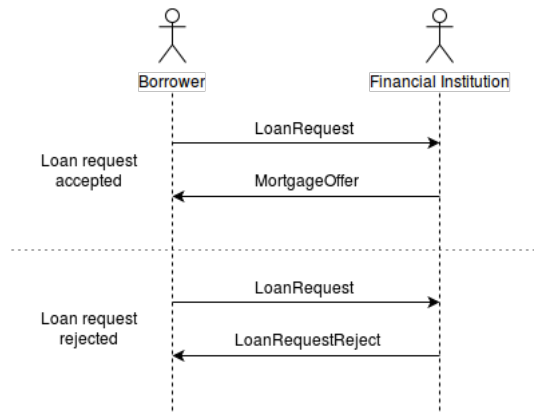


Figure 4.1: Loan request flows

### 4.2.2. Mortgage flows

When a financial institution has sent a mortgage offer to a borrower, the borrower has the option to respond by either rejecting or accepting the mortgage. When the borrower accepts the mortgage, the borrower also has to broadcast a message to other users to let them know that a new campaign has started. The mortgage flows can be found in figure 4.2.

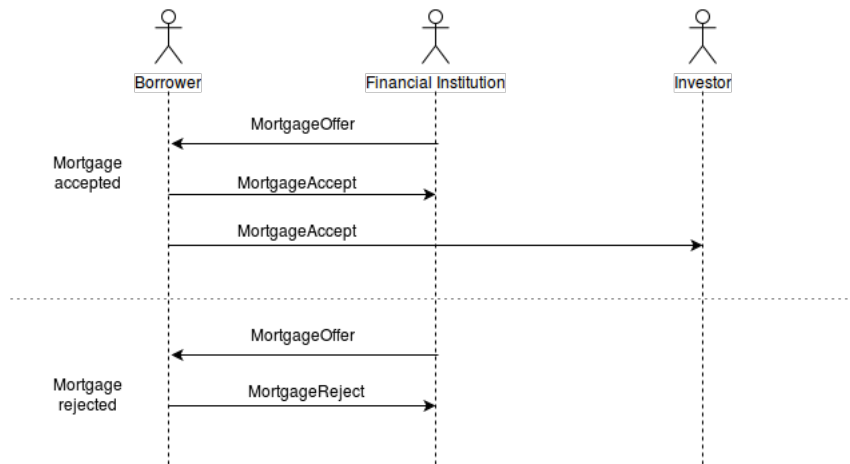


Figure 4.2: Mortgage flows

### 4.2.3. Investment flows

Once a borrower has accepted a mortgage, investors can place bids on their campaign by placing an investment offer on the borrower's campaign. The borrower can respond by either rejecting or accepting the investment. The investment flows can be found in figure 4.3.

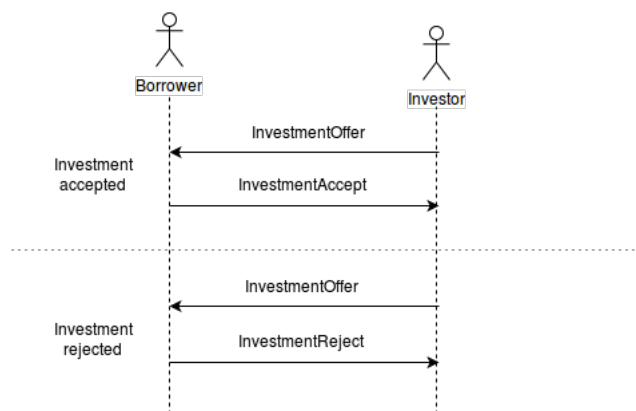


Figure 4.3: Investment flows

### 4.3. Integrating Dispersy

Dispersy lies at the heart of the decentralized mortgage marketplace. Integrating Dispersy to the market community proved to be a considerable challenge. It introduces a major paradigm change, since the program must work properly across different computers with each a different state. At the heart of Dispersy lies a simple identity and messaging system. These two systems have been fully integrated into the marketplace. In the following two subsections these integrations will be discussed.

#### 4.3.1. Dispersy and Market Identity

Dispersy implements an identity system based on public/private keys. A 'Member' is someone who is part of a Dispersy community, and is identified by a unique public key. As soon as the member is connected, a 'Candidate' is created. A 'Candidate' is the object representation of the network address of the 'Member'. The member answers the question 'who', and the candidate the question 'where'.

Having read the Dispersy documentation prior to designing the API, the decision was made to identify the users in the market using public/private pair as well. To ensure that the identity of a user is always known offline as well as online, a key is generated and used for the creation of the Dispersy member and the market user. This key pair is then saved in the local database to be retrieved for future use. This process is completely transparent to the user.

The candidate however is not saved to any persistent database, rather to a memory location in the Market API. Candidates are volatile, IP address change often, connections get dropped, etc. Thus as soon as a user goes offline, the candidates list (which associates API 'User' objects with candidates) is lost and has to be rebuilt at the next connection to the community.

#### 4.3.2. Dispersy messages sharing

Dispersy implements a variety of ways messages can be sent between peers. The method of sending a message is called a 'Distribution' in Dispersy, and the recipient of a message is determined by the 'Destination'. There are multiple implemented distributions and destinations, but for this project only two destinations and distributions will be used.

The 'CommunityDestination' specifies that a message must be sent to a (specified) maximum number of (randomly selected) online members of the Dispersy community. To send a message to specific peers the 'CandidateDestination' must be used, which expects a list of peers to whom the message must be sent.

These destinations can be reached using two distribution methods. Using the 'DirectDistribution' the message is sent directly to a peer, then discarded. This distribution is used in the market application alongside 'CandidateDestination' to handle direct messages to peers. As mention above, the 'CommunityDestination' sends the message to a list of randomly selected peers, as such to ensure that all peers receive the message, 'DirectDistribution' is not advisable in this case, as after the randomly selected peers receive the message it is discarded. This is solved using the 'FullSyncDistribution'.

The 'FullSyncDistribution' instructs Dispersy to store the message for further propagation in the future. Furthermore, newly connected peers can ask online peers to send them all messages that they missed since they were last online. This message type is used for certain messages which ought to be shared to as many

peers as possible such as open campaigns, investments, etc. This distribution method will be used alongside the 'CommunityDestination' to share all public data.

## 4.4. Protocol specification

In this section we explain the API messages that are needed for the market protocol, the messages that are needed for the Dispersy community, and the signed messages that are needed to create blocks for the blockchain.

### 4.4.1. API Messages

In this section we describe the messages that are needed for the market protocol, which is described in section 4.2. These messages are different from the Dispersy messages described in the previous section. This was to remove the amount of duplicated code since all API messages have the same payload format.

#### **LoanRequest**

A LoanRequest message is sent by a borrower to a financial institution, when they create a loan request. The LoanRequest message is an unsigned message. The payload of the message contains the objects LoanRequest, House and Profile.

#### **LoanRequestReject**

A LoanRequestRejected message is sent by a financial institution to a borrower, when it rejects a loan request. The payload of the message contains the LoanRequest object.

#### **MortgageOffer**

A MortgageOffer message is sent by a financial institution to a borrower, when it accepts a loan request. The payload of the message contains the LoanRequest and Mortgage objects.

#### **MortgageAcceptSigned**

A MortgageAcceptSigned message is sent directly by a borrower to a financial institution, when they accept the mortgage offer. The payload of the message contains the Mortgage and Campaign objects.

#### **MortgageAcceptUnsigned**

A MortgageAcceptUnsigned message is sent by a borrower to all other users when they accept a mortgage offer. The payload of the message contains the LoanRequest, Mortgage, House and Campaign objects.

#### **MortgageReject**

A MortgageReject message is sent by a borrower to a financial institution, when they reject a mortgage offer. The payload of the message contains the Mortgage object.

#### **InvestmentOffer**

An InvestmentOffer message is sent by an investor to a borrower, when they place an investment offer. The payload of the message contains the Investment and the Profile object.

#### **InvestmentAccept**

An InvestmentAccept message is sent by a borrower to an investor, when they accept the investment offer. The payload of the message contains the Investment and the BorrowersProfile object.

#### **InvestmentReject**

An InvestmentReject message is sent by a borrower to an investor, when they reject the investment offer. The payload of the message contains the Investment object.

#### **CampaignBid**

A CampaignBid message is sent by either an investor or a borrower, when they place a bid on a campaign or accept or reject a bid respectively. The payload of the message contains the Investment and the Campaign object.

#### **ModelRequest**

A ModelRequest message can be sent by a borrower, an investor and a financial institution. This message is sent when they need a model that they do not have. This is an unsigned message. The payload of the message contains the id of the requested model.



**ModelRequestResponse**

A ModelRequestResponse message can be sent by a borrower, an investor and a financial institution and is a response to the ModelRequest message. This message is sent when they have the model requested by the ModelRequest message. This is an unsigned message. The payload of the message contains the requested object.

**4.4.2. Dispersy Messages**

As described in 4.3, messages lie at the heart of Dispersy. Six messages have been defined for the mortgage market community. In Dispersy a message must always have a 'Payload', which is a class which specifies which data will be carried in the message. These payloads must also have a decoder and encoder to serialize the data for transfer over UDP. To prevent having a host of payloads, encoders and decoders effort was made to make the payloads as general as possible to enable re-use. The payloads implemented are:

**DatabaseModelPayload**

A payload with two attributes: 'fields' and 'models'. Where fields is the list of keys pertaining to Database-Models in 'models'.

**APIMessagePayload**

This payload is similar to the DatabaseModelPayload, but with an extra attribute 'request'. This request field is used to specify which API Message handler needs to be used.

**SignedConfirmPayload**

This payload holds the following fields: 'benefactor', 'beneficiary', 'agreement\_benefactor\_encoded', 'agreement\_beneficiary\_encoded', 'sequence\_number\_benefactor', 'sequence\_number\_beneficiary', 'previous\_hash\_benefactor', 'previous\_hash\_beneficiary', 'signature\_benefactor', 'signature\_beneficiary', 'time'. These hold the keys of the two signing parties, the agreements of both parties (which should be equal), the sequence number of the block in the blockchain, the hash of the previous item in the blockchain, and the signature of the message. Finally the time is saved.

**ModelRequestPayload**

A payload with a single attribute, 'model\_ids'. To request the specified models from anyone that has them in their database.

The messages using these payloads are now defined.

**model\_request**

This is a message sent to everyone online in the community using the 'ModelRequestPayload' to request for specific models.

**model\_request\_response**

As a reply to a 'model\_request', with the 'DatabaseModelPayload' carrying the requested models.

**introduce\_user**

Sent as a reply to the built-in Dispersy message to introduce oneself to other nodes (Community.on\_introduction\_response). Contains a 'DatabaseModelPayload' with the user object.

**api\_message\_community**

This is an API message sent to the entire community using the FullSyncDistribution type. FullSyncDistribution enables the gossiping of messages, which will ensure that users who log in in at a later time can get the message with a 'dispersy-sync' request. Example usage is for instance propagation of a new campaign. These messages are saved in the Dispersy database for further propagation.

**api\_message\_candidate**

This is an API message sent directly to one or more explicitly given candidates. All communication between a borrower and a bank happens through direct messages. These messages are discarded directly after being received.

**signed\_confirm**

The signed\_confirm message is specifically created to seal an agreement between two parties. Using

the 'DoubleMemberAuthentication' provided by Dispersy, it requires that both parties sign the message if they agree with the message content. The message content being either a 'Mortgage' or 'Investment'. Thus before signing the message the system checks if the offer received is consistent with what has been agreed with. Only agreements which have passed through the signed\_confirm process are treated as final and binding.

#### 4.4.3. Signed messages protocol

When the beneficiary(borrower) accepts a mortgage offer or an investment offer, the benefactor(bank or investor) will create and send out a signature request message, with only the request part filled in, to the beneficiary. This message contains the SignedConfirmPayload listed in 4.4.2, but not all fields are filled in by the benefactor. The fields 'beneficiary', 'agreement\_beneficiary', 'sequence\_number\_beneficiary', 'previous\_hash\_beneficiary' and signature\_beneficiary will be filled in by the beneficiary. We call this the response part. The benefactor persists the message and the corresponding block. A hash will be created from the message.

Upon receipt of the signature request message, the beneficiary decides whether to accept or drop the message. When it has been decided to accept the message, the beneficiary will create and sign the response part of the message, and send it back to the benefactor. The message that is being sent back is called the signature response message. The beneficiary persists the message and the corresponding block. A hash will be created from the message.

When the benefactor receives the signature response message, and the 'agreement\_benefactor' and 'agreement\_beneficiary' are the same, and there are no inconsistencies in the payload, then the benefactor will accept the message. The benefactor updates the message and the corresponding block. A new hash will be created from the message.

A sequence diagram of the signed messages protocol can be found in figure 4.4

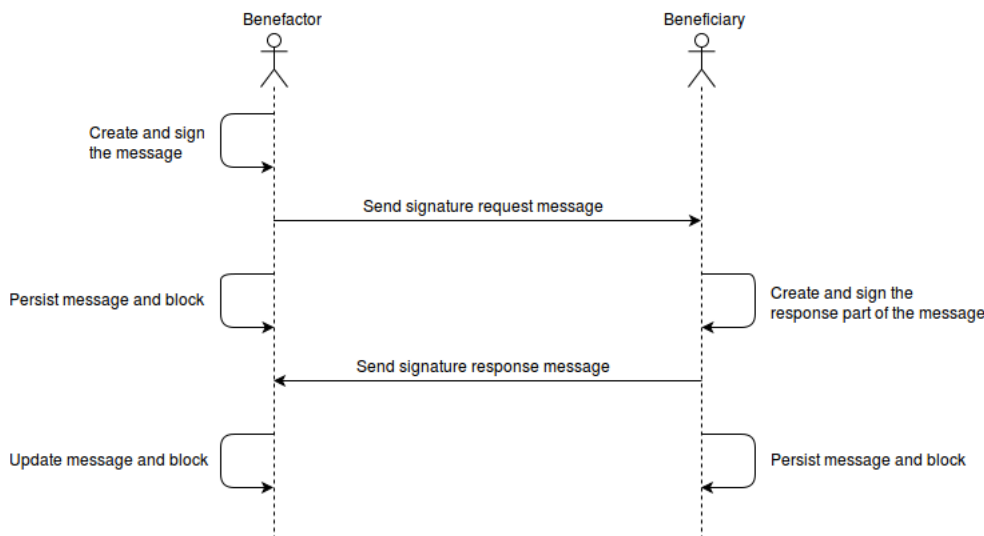


Figure 4.4: Loan request protocols

## 4.5. Blockchain

To ensure that no one can lie about an agreement they have made, a blockchain is used. We made our own blockchain, inspired by the MultiChain [38], as we need it to save specific information relevant to our market. Every block in the blockchain stands for an agreement between two users of the market. When any of these blocks are being tampered with, the blockchain will become invalid. Therefore a blockchain is a good measure to ensure more trust into the system.

The blockchain consists of individual blocks. Every block contains the following fields: 'benefactor', 'beneficiary', 'agreement\_benefactor', 'agreement\_beneficiary', 'sequence\_number\_benefactor', 'sequence\_number\_beneficiary', 'previous\_hash\_benefactor', 'previous\_hash\_beneficiary', 'signature\_benefactor', 'signature\_beneficiary', 'time', 'hash\_block', 'previous\_hash' and 'sequence\_number'. The last three fields are only visible to the owner of the

blockchain the block is added to, and are different for every user. The other fields are visible to both users that have the block in their blockchain and these fields are the same fields in the SignedConfirmPayload listed in 4.4.2.

The blocks of a blockchain are linked through the hash\_block and the previous\_hash. The hash\_block contains the hash of the current block and the previous\_hash contains the hash\_block of the block before the current block. This way, a chain of blocks are linked to each other from the very first block to the last block. Another way to order the blocks to get the whole chain is by the block's sequence number. By using the sequence number of a block, we do not need to traverse through the hashes to get the chain. A graphical representation of a part of a local blockchain can be found in figure 4.5. In this figure it can be seen how the blocks are linked to each other.

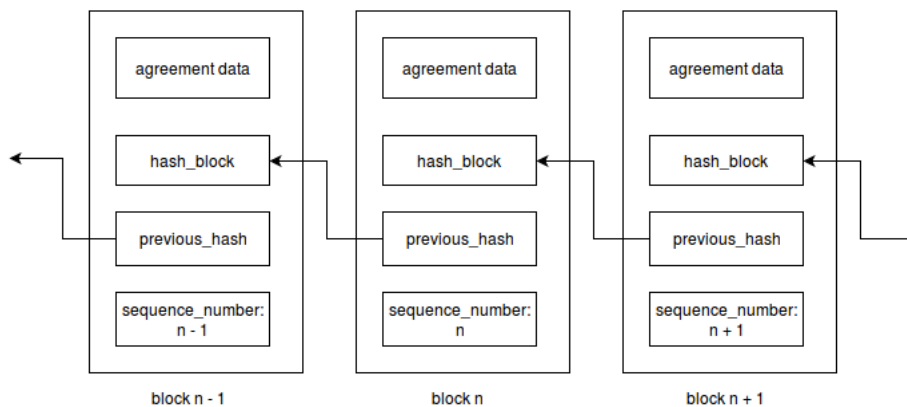


Figure 4.5: A chain of blocks in a local blockchain

At the moment, the blockchain of every user is local. This is why the three fields 'hash\_block', 'previous\_hash' and 'sequence\_number' are needed to form the local blockchain. If the local blockchain of a user should ever be synced with a blockchain of another user, it should not be a problem. In figure 4.6 it can be seen how the blocks are linked to each other when four blockchains of four users are synced. Colors are added to make it more clear.

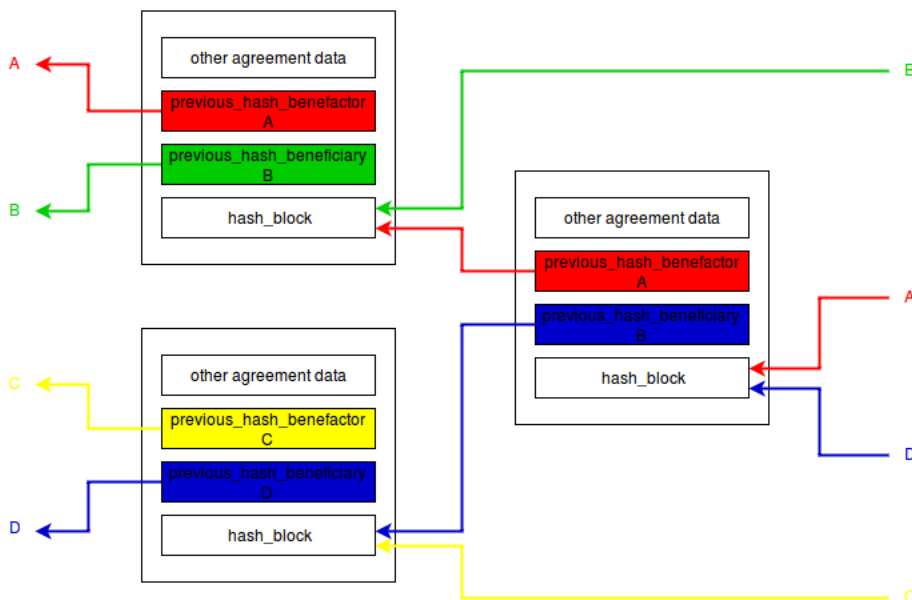


Figure 4.6: Part of a blockchain with 4 users

## 4.6. Document Sharing

Dispersy is a great platform to host the distributed market. The program uses the dispersy message packages to transfer different payloads between users. While that works very well for small payloads, its maximum payload size can be too small to hold a document. While theoretically it is still possible to send larger documents over dispersy it seemed more practical to use a protocol like TFTP [29] to transfer the documents.

To do transfer the documents the program uses a module named *tftpy* [30]. With this module files can easily be uploaded to, or downloaded from a server. In this case we want to upload the documents of a borrower to the banks that were chosen during the placement of a loan request.

For simplicity, we make the assumption that the banks are always online and are hosting a TFTP-server and we also make the assumption that every user needs to upload the same set of documents. Furthermore, we assume that when the transfer of the documents fails we will be able to try to transfer them again at a later date.

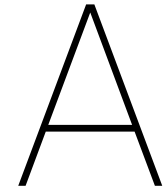
When the user places a loan request, all files are added to a queue and uploaded to all chosen banks one by one. If something goes wrong with transferring the documents, the jobs are saved to be resent on a later date.

# Bibliography

- [1] Gedragscode Hypothecaire Financieringen / Code Of Conduct For Mortgage Loans. URL <https://www.nvb.nl/publicaties-standpunten/publicaties/1671/gedragscode-hypothecaire-financieringen-code-of-conduct-for-mortgage-loans.html>.
- [2] D&B Rating. URL <http://www.dnb-nederland.nl/dnb-data/rating>.
- [3] ABN AMRO. URL <https://www.abnamro.nl/>.
- [4] Blandlord. URL <https://blandlord.com/>.
- [5] Blockchains: The great chain of being sure about things, . URL <http://www.economist.com/news/briefing/21677228-technology-behind-bitcoin-lets-people-who-do-not-know-or-trust-each-other-but>
- [6] Blockchain Lab, . URL <http://www.blockchain-lab.org/>.
- [7] Bouwsteen.org. URL <http://www.bouwsteen.info/>.
- [8] Collin Crowdfund. URL <https://www.collincrowdfund.nl/>.
- [9] CrowdAboutNow. URL <https://crowdaboutnow.nl/>.
- [10] Dispersy. URL <https://github.com/Tribler/dispersy>.
- [11] De Nederlandsche Bank / Dutch National Bank. URL <https://www.dnb.nl/>.
- [12] Fundrise. URL <https://fundrise.com/>.
- [13] geldvoorelkaar. URL <http://www.geldvoorelkaar.nl/>.
- [14] GitHub, . URL <https://github.com/>.
- [15] Project Description issue on Github, . URL <https://github.com/Tribler/tribler/issues/2606>.
- [16] Gradefix. URL <https://gradefix.com/>.
- [17] Greencrowd. URL <https://greencrowd.nl/>.
- [18] Jenkins. URL <https://jenkins.io/>.
- [19] Jungo. URL <https://jungo.nl>.
- [20] Kapitaal op Maat. URL <https://www.kapitaalopmaat.nl/>.
- [21] Kamer van Koophandel/ Chamber of Commerce. URL <https://www.kvk.nl/>.
- [22] Loan-To-Value Ratio - LTV Ratio. URL <http://www.investopedia.com/terms/l/loantovalue.asp>.
- [23] MoneYou. URL <https://www.moneyou.nl>.
- [24] MoSCoW Prioritisation. URL <https://www.agilebusiness.org/content/moscow-prioritisation>.
- [25] NHP Soorten hypotheek. URL <https://nhp.nl/page/?permalink=hypotheek/informatie/hypotheekvormen/soorten-hypotheek&piranha-culture=nl>.
- [26] RealtyShares. URL <https://www.realtyshares.com/>.
- [27] Koophuis - Hypotheek - Maximale hoogte hypotheek. URL <https://www.rijksoverheid.nl/onderwerpen/koopwoning/inhoud/hypotheek>.

- [28] Scrum Alliance. URL <https://www.scrumalliance.org/>.
- [29] RFC 1350 - The TFTP Protocol (Revision 2), . URL <https://tools.ietf.org/html/rfc1350>.
- [30] TFTPpy - A Pure Python TFTP Protocol Implementation, . URL <http://tftpy.sourceforge.net/>.
- [31] Travis CI. URL <https://travis-ci.org/>.
- [32] About Tribler, . URL <https://www.tribler.org/about.html>.
- [33] Tribler GitHub Wiki, . URL <https://github.com/Tribler/tribler/wiki>.
- [34] TU Delft and ABN AMRO to collaborate on the development of blockchain applications. URL <http://www.tudelft.nl/en/current/latest-news/article/detail/samenwerking-tu-delft-en-abn-amro-voor-ontwikkeling-blockchain-toepassingen/>.
- [35] Upstart. URL <https://www.upstart.com/>.
- [36] Vivek Kapoor, Vivek Sonny Abraham, and Ramesh Singh. Elliptic curve cryptography. *Ubiquity*, 2008 (May):7, 2008.
- [37] J. Winter M.J.G. Olsthoorn. Decentral market: self-regulating electronic market. Master's thesis, Delft University of Technology, 2016.
- [38] S.D. Norberhuis. MultiChain: A cybocurrency for cooperation. Master's thesis, Delft University of Technology, 2015.
- [39] RTL Z / Erik Rezelman. DNB-topman Klaas Knot: aftrek hypotheekrente versneld afbouwen. URL <http://www.rtlnieuws.nl/geld-en-werk/dnb-topman-klaas-knot-aftrek-hypotheekrente-versneld-afbouwen>.
- [40] Manoj Singh. The 2007-08 Financial Crisis In Review. URL <http://www.investopedia.com/articles/economics/09/financial-crisis-review.asp>.

# **Appendices**



# Project description: Open market for real-estate crowdsourcing

Market platforms such as AirBnB, eBay, Uber, and Blandlord.com bring supply and demand together. Based on our ongoing "blockchain-regulated markets" we now explore usage in various domains. This issue addresses an open market for crowdsourcing in real-estate.[15]

Outcome of initial brainstorm sketches:

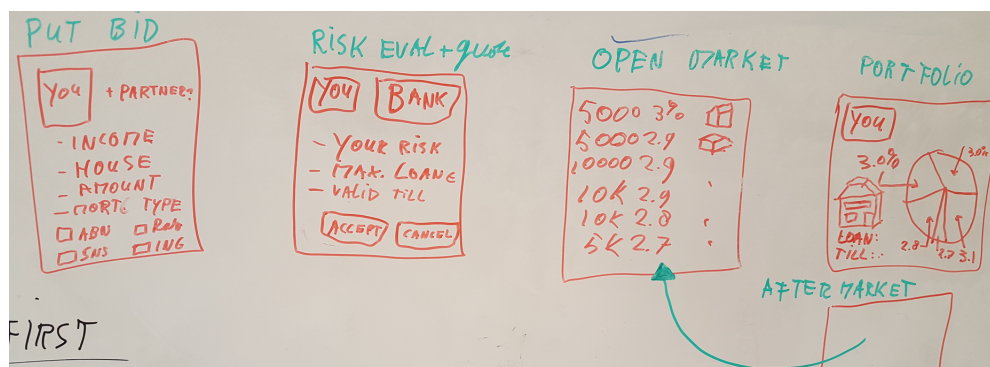


Figure A.1: Brainstorm sketches

## Sprints:

- Week 1-2: Write-up of context and understanding of problem, understand related initiatives. Describe at least 10 initiatives. Learn Python.
- Week 3-4: Apply Python. Visualize the concept using PyQt designer with 4 operational screens. Feedback round from bank.
- Week 5-6:
- Week 7-8:

## Roadmap:

- Feb/Jan: we have 3 components ready in draft: operational market + GUI implemented in QT + business case
- till April: define API between market and business logic, create business logic core, integrate the market, business logic, and GUI.
- April: present MVP results



Various markets for real-estate already exist, such as:

**Blandlord is gestart in Rotterdam.**

Rotterdam is een ideale stad om huizen te verhuren. De universiteit en vele internationale bedrijven maken dat er grote vraag is naar woonruimte in het middensegment. Het eerste huis is nu beschikbaar in de verkoop. Het appartement ligt in Noord nabij het centrum en is per 1 oktober verhuurd.

**Schieweg, Rotterdam**

€ 119,000 Koopsom      4.01% Rental income

[View details](#)

4.0%

Op de eerste verdieping gelegen ruim opgezet 2-...  
Rotterdam, Noord

VHPN

already sold 52%

Heb je een vraag? Stel hem hier

Hoe werkt het?

Figure A.2: Blandlord is gestart in Rotterdam


Plus an operational crowdsourcing market, running locally which TUDelft indirectly has access to:

Crowdfunding project Wilsta Prefableidingen - Kapitaal Op Maat - Chromium

Crowdfunding project x

https://www.kapitaalopmaat.nl/Wilsta-Prefab-Leidingen

Investeren Lenen Hoe werkt het? Projecten Inloggen Registreren



Deel dit project [f Share](#) [Tweet](#) [Delen](#) [G+ plus 1](#)

Het project [Investeringen \(119\)](#) [Financieel](#) [Zekerheden](#) [Reacties \(0\)](#) [Updates \(0\)](#)

**Wilsta Prefab-leidingen en handelsonderneming is een jong, flexibel en snelgroeiend bedrijf. Wij zijn gespecialiseerd in het vervaardigen en leveren van prefab-leidingen in kunststof (PVC, PE, PP, en PPC), koper, RVS en staal. Wilsta levert maatwerk voor de installateur op het gebied van riolering, mechanische ventilatie en leidingenwerk. Ook voor medische gassen leveren wij de benodigde leidingdelen.**

**Het ontstaan van Wilsta**  
 Wilsta prefab-leidingen BV is opgericht in 2008 door Wilma Kollenburg met steun en advies van haar man Stan Witlox. Samen runnen zij het familiebedrijf. Stan Witlox is de derde generatie in het installatie vak. Vanaf het begin van de relatie hielp Wilma al mee (veelal in de avonduren) om in de werkplaats de leidingdelen voor te bewerken. In 1992 zijn Stan en Wilma in het huwelijk getreden en hebben zij het installatiebedrijf in 1994 van de ouders van Stan overgenomen. Met het installatiebedrijf werden voornamelijk werkzaamheden uitgevoerd die geschikt waren om innovatief en creatief te werk te gaan. Het installatiebedrijf is mede doordat Stan door de zware werkzaamheden rugproblemen kreeg gestopt. Wilma is hierdoor gestart met Wilsta. Stan functioneert nu als opzichter.

**Wat levert Wilsta?**  
 Wilsta levert maatwerk voor de installateur op het gebied van riolering, mechanische ventilatie en leidingenwerk. Ook voor medische gassen levert Wilsta de benodigde leidingdelen. Alle prefab leidingen worden vanuit de eigen werkplaats vervaardigd en door eigen personeel geleverd en desgewenst gemonteerd. Wilsta is gespecialiseerd in fusiotherm/aquatherm leidingsystemen, PE leidingen en HDPE leidingen. Wij monteren tevens leidingsystemen volgens het perssysteem. Wij werken met de nieuwste machines en werken nauw samen met fabrikanten en engineers om leidingsystemen te verbeteren. Wilsta is ook actief voor bedrijven in de weg- en waterbouw. Voor deze bedrijven maken wij leidingsystemen uitgevoerd in Pe en Hdpe. Speciaal voor deze branche heeft Wilsta prefableidingen graafmachines, elektrolas en spiegellasmachines.

**Specifiek werk en ervaring**  
 Door de speciale manier van werken hebben importeurs van leidingen de weg naar Wilsta gevonden om deze te gaan prefabriceren. Deze leidingen worden in de eigen werkplaats gemaakt. Wilsta monteert deze leidingen "op het werk" en is een veel gevraagde partner geworden voor grote installatiebedrijven. Hierdoor is Wilsta hard gegroeid en heeft tot op heden een indrukwekkende referentie lijst opgebouwd. Wij werken uitsluitend met de beste materialen.

**100% Gefinancierd!**  
 Met dank aan **111** investeerders!

**Investeringsrisico**  
 A- B+ B B- C+ **C** C-  
 laag hoog


**Financieringsduur**  
**60** Maanden

**Rente**  
**8** %

**Reeds geïnvesteerd**  
**€ 150.000**  
 van de € 150.000

**Doel van investering**  
 Financiering werkkapitaal

**Maak kennis met...**  
 W.M.M.M Witlox-Kollenburg



Wilsta prefableidingen is opgericht in 2008 door Wilma Kollenburg met steun en advies van haar man Stan Witlox. Samen runnen zij het familiebedrijf.

**Geen project van uw keuze?**  
 Mail mij bij nieuwe projecten.

E-mailadres  [Mail mij](#)

Figure A.3: Kapitaal op Maat

B

Mortgage

### Welke documenten heb je nodig voor een hypotheek?

Als je een hypotheek afsluit, heb je nogal wat documenten nodig. Welke documenten je nodig hebt voor een hypotheek, ligt aan de fase van het hypotheekadviestraject. En aan jouw situatie. Sommige documenten heb je al nodig bij het oriëntatiegesprek. Andere documenten heb je pas nodig bij het afsluiten van je hypotheek.

### Documenten voor hypotheek per fase

In onderstaande tabel zie je welke documenten je voor je hypotheek nodig hebt in welke fase. Dit zijn de 4 fasen van het hypotheekadviestraject waarin verschillende documenten nodig zijn:

1. Oriëntatie: berekening maximale hypotheek en voorbeeld maandlasten.
2. Hypotheekadvies: hypotheek samenstellen en producten vergelijken.
3. Hypotheekofferte aanvragen.
4. Hypotheek afsluiten: getekende hypotheekofferte naar de geldverstrekker.

### Documenten voor hypotheek per situatie

In onderstaande tabel zie je welke documenten je voor je hypotheek nodig hebt in welke situatie. De algemene documenten heb je altijd nodig. De overige documenten alleen als die situatie voor jou geldt.

Alle documenten mogen een scan of kopie zijn, tenzij er "(origineel)" achter staat.

Identificatie	1	2	3	4
Paspoort of ID-kaart	x	x	x	x
Paspoort of ID-kaart (origineel)*	x	-	-	-
* Origineel moet gecontroleerd worden.				
<b>Werk</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
Loonstrook	x	x	x	x
<a href="#">Werkgeversverklaring</a>	x	x	x	-
Jaaropgaven afgelopen 3 jaar *	x	x	x	x
Arbeidsovereenkomst **	-	-	-	x
Rekeningafschrift met bijschrijving netto salaris	-	-	-	x
Arbeidsverleden via <a href="#">Mijn UWV</a>	x	x	x	-

\* Als je geen vast contract én geen intentieverklaring hebt.

\*\* Als ingangsdatum minder dan 6 maanden geleden is.

<b>Uitgaven</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
Budgetoverzicht *	-	x	-	-

\* Bijvoorbeeld via [Nibud Persoonlijk Budgetadvies](#).

<b>Vermogen</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
Afschrift spaarrekening(en)	-	x	x	x
Opgave beleggingen	-	x	x	x

<b>Vorzieningen werkgever</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
Uniform Pensioen Overzicht (UPO)	-	x	x	x
Opgave <a href="#">MijnPensioenOverzicht.nl</a>	-	x	x	x
Verzekering arbeidsongeschiktheid	-	x	x	x

<b>Woning</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
Getekende koopovereenkomst	-	-	x	x
Taxatierapport *	-	x	x	-

\* PDF van een gevalideerd taxatierapport.

<b>Verbouwing</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<a href="#">Specificatie verbouwingskosten</a>	-	x	x	-

<b>Schenking ouders</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
Getekende overeenkomst	-	-	-	x
Paspoort of id-kaart ouders	-	-	-	x

<b>Lening ouders</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
Getekende overeenkomst	-	-	-	x
Paspoort of id-kaart ouders	-	-	-	x

<b>Lening</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
Overeenkomst	-	x	x	x
Bankafschrift met maandbedrag	-	x	x	x
Bewijs aflossing *	-	-	-	x
Bewijs afmelding BKR *	-	-	-	x
Bewijs eigen geld *	-	-	-	x

\* Als je jouw lening of krediet moet opzeggen voor je hypotheek.

<b>Studielening</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
Opgave DUO begin schuld	-	x	x	x
Opgave DUO huidige schuld	-	x	x	x

<b>Gescheiden</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
Convenant	-	x	x	x
Inschrijving gemeente	-	x	x	x
Akte van verdeling *	-	x	x	x

Vonnis rechtbank **	-	x	x	x
---------------------	---	---	---	---

Inschrijving vonnis **	-	x	x	x
------------------------	---	---	---	---

\* Als je partner de woning overneemt.

\*\* Als jouw scheiding via de rechtbank gaat.

<b>Flexwerker</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
-------------------	----------	----------	----------	----------

Jaaropgaven afgelopen 3 jaar	-	x	x	x
------------------------------	---	---	---	---

Arbeidsovereenkomst	-	x	x	x
---------------------	---	---	---	---

<b>Zelfstandig ondernemer</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
-------------------------------	----------	----------	----------	----------

Uittreksel KvK	-	-	x	x
----------------	---	---	---	---

Jaarcijfers 3 jaar	x	x	x	x
--------------------	---	---	---	---

Aangiftes IB 3 jaar	x	x	x	x
---------------------	---	---	---	---

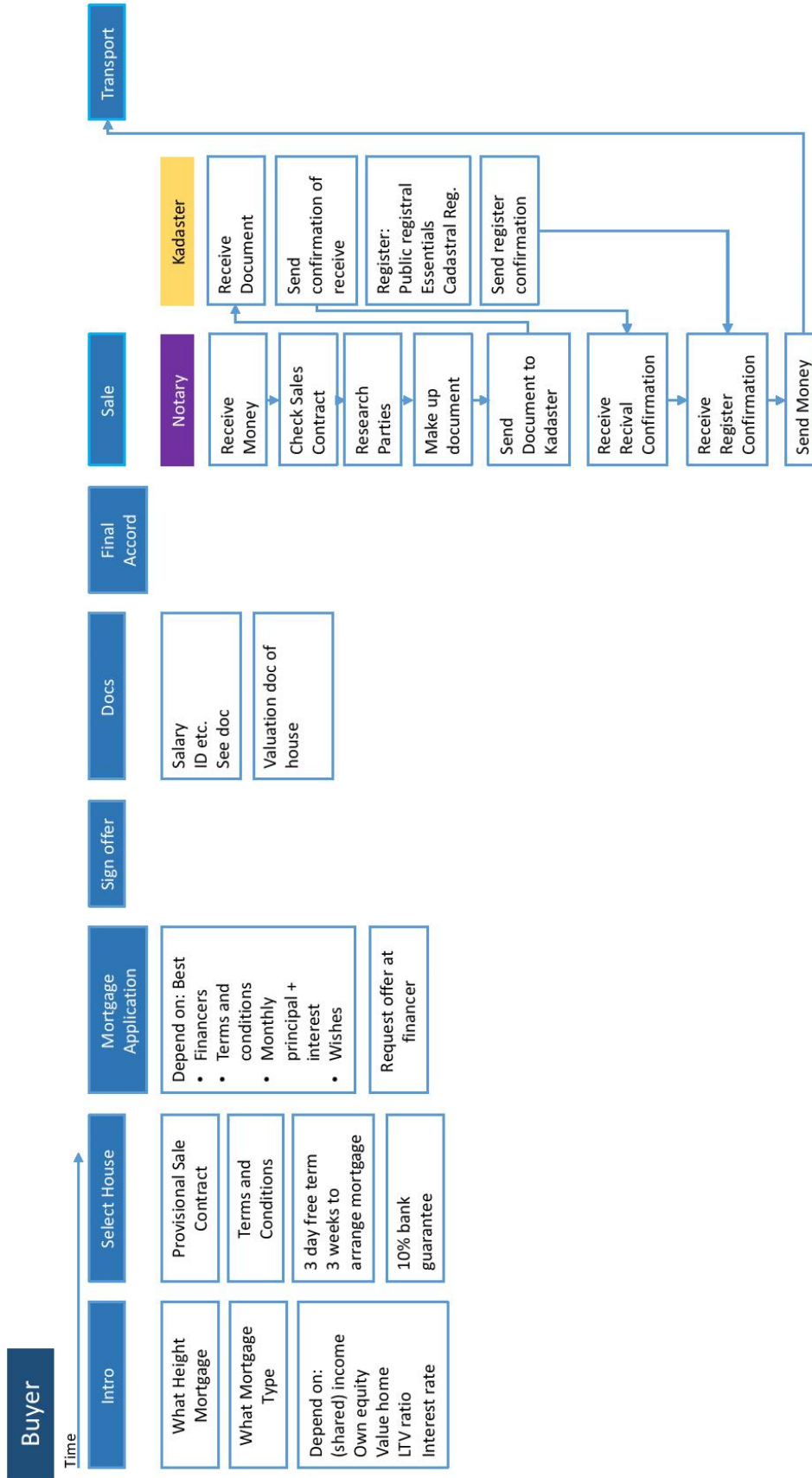
Aanslagen IB 3 jaar	x	x	x	x
---------------------	---	---	---	---

Voorlopige cijfers	x	x	x	x
--------------------	---	---	---	---

<b>Uitkering</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
------------------	----------	----------	----------	----------

Specificatie uitkering	x	x	x	x
------------------------	---	---	---	---

Jaaropgaaf	x	x	x	x
------------	---	---	---	---





C

API

## Market API

```
class market.api.api.MarketAPI(database) [source]
```

Create a MarketAPI object.

The constructor requires one variable, the *Database* used for storage.

```
accept_investment_offer(user, payload) [source]
```

Accept an investment offer for the given user.

The payload dictionary has the following composition

Key	Description
investment_id	The id of the investment

**Parameters:**

- **user** (`User`) – The user accepting an investment offer.
- **payload** (*dict*) – The payload containing the data for the `Investment`, as described above.

**Returns:** Returns True if successful, False otherwise.

**Return type:** bool

**Raise:** AssertionError if the user does not have a campaign assigned.

```
accept_loan_request(bank, payload) [source]
```

Have the loan request passed by the payload be accepted by the bank calling the function.

The payload is as follows:

Key	Description
request_id	The <code>LoanRequest</code> id
amount	The amount the bank is willing to finance
mortgage_type	The mortgage type: 1 = linear, 2 = fixed-rate
interest_rate	The interest rate to be paid over the financed amount (float)
default_rate	The default rate (float)

Key	Description
max_invest_rate	The maximum investment interest rate (float)
duration	The duration of the mortgage
investors	List of initial investors, can be empty.

**Parameters:**

- **bank** (`User`) – The bank accepting the loan request.
- **payload** (`dict`) – The payload containing the data for the `Mortgage`, as described above.

**Returns:** Returns the loan request and the mortgage objects, or None if an error occurs.

**Return type:** tuple(`LoanRequest`, `Mortgage`) or None

**accept\_mortgage\_offer**(`user`, `payload`) [\[source\]](#)

Accept a mortgage offer for the given user.

This action automatically rejects all other mortgage offers.

The payload dictionary has the following composition

Key	Description
mortgage_id	The id of the mortgage

**Parameters:**

- **user** (`User`) – The user accepting a mortgage offer
- **payload** (`dict`) – The payload containing the data for the `Mortgage`, as described above.

**Returns:** Returns True if successful, False otherwise.

**Return type:** bool

**create\_campaign**(`user`, `mortgage`, `loan_request`) [\[source\]](#)

Create a funding campaign with crowdfunding goal the difference between the price of the house and the amount requested from the bank. #TODO: Should it be more flexible?

**Parameters:**

- `user (User)` – The `User` for who the mortgage is being made
- `mortgage` – The `Mortgage` pertaining to the house being financed
- `loan_request (LoanRequest)` – The `LoanRequest` created prior to the mortgage being accepted.

**Type:** mortgage: `Mortgage`

**Returns:** True if created, None otherwise.

**Return type:** bool or None

### `create_loan_request(user, payload)` [\[source\]](#)

Create a Loan request for the given user using the payload provided.

The payload dictionary has the following composition

Key	Description
postal_code	The postal code of the house that is the target of the mortgage
house_number	The house number of the house that is the target of the mortgage
price	The total price of the house
mortgage_type	The mortgage type: 1 = linear, 2 = fixed-rate
banks	List of banks the request should be sent to
description	Free text (unicode)
amount_wanted	The amount the borrower wants financed

**Parameters:**

- `user (User)` – The user creating a loan request
- `payload (dict)` – The payload containing the data for the `House` and `LoanRequest`, as described above.

**Returns:** The loan request object if succesful or False

**Return type:** `LoanRequest` or False

### `create_profile(user, payload)` [\[source\]](#)

Creates a new profile and saves it to the database. The profile can either be a normal Profile or a BorrowersProfile, depending on the role given in the payload. Overwrites the old profile. The role 'FINANCIAL\_INSTITUTION' can not have a profile, but the role will be set. Thus the function will return `True`.

The payload contains the following data. If the role is *1* for *BORROWER* then the last three fields must also be pushed.

Key	Description
role	The role id uit of the following tuple: ('NONE', 'BORROWER', 'INVE'
first_name	The user's first name
last_name	The user's last name
email	The user's email address
iban	The user's IBAN
phonenumber	The user's phone number
current_postalcode	The user's current postal code
current_housenumber	The user's current house number
documents_list	A list of documents

- Parameters:**
- `user (User)` – The user for whom a profile has to be made
  - `payload (dict)` – The payload containing the data for the profile, as described above.

:return The Profile or True if a bank role was set or False if the payload is malformed  
:rtype `Profile` or True or False

### `create_user()` [\[source\]](#)

Create a dispersy user by generating a key public/private pair.

Returns None if the user creation process fails.

**Returns:** A tuple with the User object, the public key and the private key. The keys encoded in HEX.

**Return type:** (`User`, Public, Private) or None

### `get_role(user)` [\[source\]](#)

Get the role of the user from the database.

**Parameters:** `user` – The `User` whose role you want to retrieve

**Returns:** : Returns the role or None.

**Return type:** `User` or `None`

### `load_all_loan_requests(user)` [\[source\]](#)

Display all pending loan requests for the specific bank

**Parameters:** `user (User)` – The bank `User`

**Returns:** A list of the :any: 'LoanRequest's if there are any, False otherwise.

**Return type:** list or False

#### `load_bids(payload)` [\[source\]](#)

Returns a list of all bids on the selected campaign.

The payload dictionary has the following composition

Key	Description
mortgage_id	The id of the selected mortgage

**Parameters:** `payload (dict)` – The payload containing the data for the `Investment`, as described above.

**Returns:** A list of :any: 'Investment' objects.

**Return type:** list

#### `load_borrowers_loans(user)` [\[source\]](#)

Get the borrower's current active loans (funding goal has been reached) or the not yet active loans (funding goal has not been reached yet) :param user: User-object, in this case the user has the role of a borrower :return: list of the loans, containing either the current active loans or the not yet active loans

#### `load_borrowers_offers(user)` [\[source\]](#)

Get all the borrower's offers(mortgage offers or loan offers) from the database. :param user: User-object, in this case the user has the role of a borrower :return: list of offers, containing either mortgage offers or investment offers :rtype: list

#### `load_investments(user)` [\[source\]](#)

Get the current investments list and the pending investments list from the database.

**Parameters:** `user (User)` – The user whose investments need to be retrieved.

**Returns:** A tuple containing the list of current and pending investments

**Return type:** tuple(CurrentInvestments, PendingInvestments)

#### `load_open_market()` [\[source\]](#)

Returns a list of all mortgages who have an active campaign going on.

**Returns:** A list of `Mortgage` objects.

**Return type:** list

#### `load_profile(user)` [\[source\]](#)

Load the given users profile.

Depending on the user's role, it will return a `Profile` for an investor or a `BorrowersProfile` for a borrower. None for a financial institution

**Parameters:** `user (User)` – The user whose profile has to be loaded.

**Returns:** `Profile` or `BorrowersProfile` or None

#### `load_single_loan_request(payload)` [\[source\]](#)

Display the selected pending loan request

**Returns:** The :any: 'LoanRequest's

**Return type:** `LoanRequest`

#### `login_user(private_key)` [\[source\]](#)

Login a user by generating the public key from the private key supplied, and searching the user object in the database using the generated key.

**Parameters:** `private_key (str)` – The private key of the user encoded in HEX.

**Returns:** The logged in User if successful, None otherwise.

#### `place_loan_offer(investor, payload)` [\[source\]](#)

Create a loan offer by an investor and save it to the database. This offer will always be created with status as 'PENDING' as the borrower involved is the only one allowed to change the status of the loan offer.

The payload contains the following data:

Key	Description
amount	The amount being invested
duration	The duration of the loan in months
interest_rate	The interest due to be paid over the loan
mortgage_id	The id of the mortgage being financed

- Parameters:**
- **investor** (`User`) – The investor wishing to invest in a mortgage by placing a loan offer.
  - **payload** (`dict`) – The payload containing the data for the `Investment`, as described above.

**Returns:** The loan offer if successful or False.

**Return type:** `Investment` or False

### `reject_investment_offer(user, payload)` [\[source\]](#)

Decline an investment offer for the given user.

The payload dictionary has the following composition

Key	Description
investment_id	The id of the investment

- Parameters:**
- **user** (`User`) – The user rejecting an investment offer.
  - **payload** (`dict`) – The payload containing the data for the `Investment`, as described above.

**Returns:** Returns True if successful, False otherwise.

**Return type:** bool

### `reject_loan_request(user, payload)` [\[source\]](#)

Decline an investment offer for the given user.

The payload dictionary has the following composition

Key	Description
request_id	The id of the loan request

- Parameters:**
- **user** (`User`) – The bank rejecting a loan request.
  - **payload** (`dict`) – The payload containing the data for the `LoanRequest`, as described above.

**Returns:** Returns the rejected :any: 'LoanRequest' if successful, None otherwise.

**Return type:** `LoanRequest` or None



**reject\_mortgage\_offer**(*user, payload*) [\[source\]](#)

Decline a mortgage offer for the given user.

The payload dictionary has the following composition

Key	Description
mortgage_id	The id of the mortgage

**Parameters:**

- **user** (`User`) – The user rejecting a mortgage offer
- **payload** (*dict*) – The payload containing the data for the `Mortgage`, as described above.

**Returns:** Returns True if successful, False otherwise.

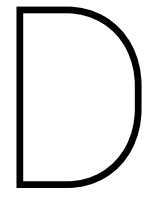
**Return type:** bool

**resell\_investment**() [\[source\]](#)

Resell an investment

Not implemented yet.

**Raise:** `NotImplementedError`



First version of the core screens

## Pending Loan Requests

### Loan Request

#### Personal Information

First Name

Last Name

Address

Telephone Number

Current Risk Rating

Email Address

Documents

#### Mortgage Request Information

Property Address

Requested Loan Amount

Mortgage Type

Property Value

#### Make Offer

Loan Amount Offered (€)

Interest (%)

Default Rate

Loan Duration (months)

Reject

Accept



## Borrower's Portfolio

### Overview ongoing loans

Loan Amount (€)	Interest (%)	Default (%)	Duration (month)	Type

### Open offers

Loan Amount (€)	Interest (%)	Default (%)	Duration (month)	Type