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 `LoanRequest` 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 invesment

Not implemented yet.

Raise: *NotImplementedError*