Documentação Arquitetónica da Online Trading Platform

No seguinte documento, “Online Trading Platform” estará abreviado como OTP.

Versão 2

29 de Novembro, 2017

Ana Rita Marques, A74218

Arquiteturas de Software
Engenharia de Sistemas de Software

Mestrado Integrado em Engenharia Informática
Universidade do Minho

1. Introdução e Objetivos

OTP é um projeto que permite **investidores** e ***traders*** abrir, fechar e gerir posições no mercado financeiro, podendo envolver a compra e venda de ativos financeiros. Os ativos financeiros poderão ser:

- Ações;
- *Commodities* (ouro, prata, petróleo);

Esta plataforma será oferecida gratuitamente aos seus utilizadores.

1. 1. Requirements Overview

O principal objetivo da *OTP* é permitir a gestão de posições no mercado financeiro.

Além disso, há um conjunto de requisitos que a plataforma deverá satisfazer.

1. 1. 1. Requisitos Funcionais

Os utilizadores da plataforma deverão poder:

- Visualizar a lista de ativos a serem negociados via CFDs, assim como os seus valores;
- Abrir conta de *trader*/investidor com um plafond inicial para negociação/investimento;
- Abrir posições (CFDs) sobre os ativos disponíveis.
- Monitorizar em 'tempo real' o seu portfólio de CFDs e para cada ativo visualizar o valor atual do(s) ativo(s) adquirido(s).

A plataforma deverá manter um ilimitado número de ativos, assim como permitir um ilimitado número de *traders*/investidores

1. 1. 2. Requisitos não funcionais

Tab. 1. Requisitos não funcionais

Nr.	Qualidade	Motivação
1	Precisão	O sistema guarda e mostra informações e cálculos precisos.
2	Facilidade de	Os utilizadores devem conseguir utilizar a

	aprendizagem	aplicação com algumas ajudas por parte da equipa de desenvolvimento
3	Modificabilidade	Quando necessárias alterações no código, é fácil perceber em que zona do código devem ser feitas.

1. 2. Stakeholders

A seguinte tabela contém os principais intervenientes desta aplicação.

Tab. 2. Stakeholders

Papel	Objetivos
Trader/Investidor	Procuram uma plataforma para a gestão de ativos financeiros.
Developer/Software Architect	Desenvolve a plataforma de *trading*.

2. Restrições

2. 1. Restrições Técnicas

Tab. 3. Restrições técnicas

	Restrição	**Background e/ou motivação**
TC1	Implementação em Python	Desenvolvimento sob versão 3.4.0 do Python.
TC2	Software de terceiros disponível gratuitamente sob uma licença de código aberto compatível	O software externo adicionado à solução deve ser gratuito e disponível gratuitamente. O desenvolvedor ou arquiteto interessado deve poder verificar as fontes, compilar e executar o aplicativo sem problemas para compilar ou instalar dependências.

TC3	Desenvolvimento independentemente do OS	A aplicação deve ser compilável nos sistemas operativos Mac OS X, Linux e Windows.

2. 2. Restrições Organizacionais

Tab. 4. Restrições organizacionais

	Restrição	**Background e/ou motivação**
OC1	Equipa	Ana Rita Marques
OC2	Horário	Desenvolvimento começou no início de Outubro e terminou a 29 de Novembro
OC3	Modelo de processo	arc42 é usado para documentar a arquitetura.
OC4	Ferramentas de desenvolvimento	Criação do código fonte Python no PyCharm.
OC6	Publicado sob licença Open Source	O código e a documentação devem ser publicados como Open Source.

2. 3. Convenções

Tab. 5. Convenções

	Convenções	**Background e/ou motivação**
C1	Documentação de arquitetura	Estrutura baseada na versão inglesa arc42-Template, version 6.5.
C2	Convenções de programação	O projeto usa convenções de programação em Python.
C3	Linguagem	Português.

3. Context

3. 1. Business Context

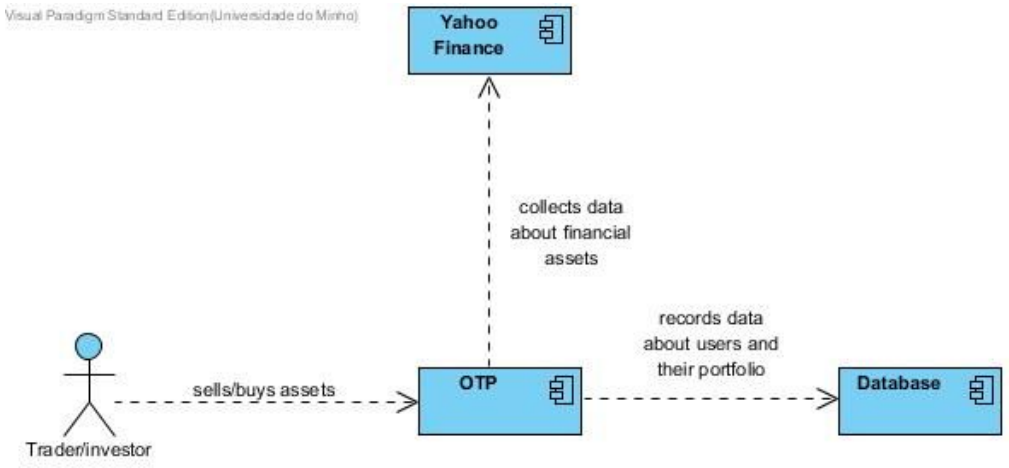


Fig. 1. Business Context Diagram

Tab. 6. Componentes

Componente	**Descrição**
Trader/investidor	O trader/investidor usa a OTP para vender, comprar e gerir ativos financeiros, mantidos no seu portfólio.
Yahoo Finance	Yahoo Finance é a data source da plataforma. Fornece os valores dos ativos.
Database	As informações sobre os ativos, utilizadores e seus portefólios estarão armazenados numa base de dados.

3. 2. Contexto de Implementação

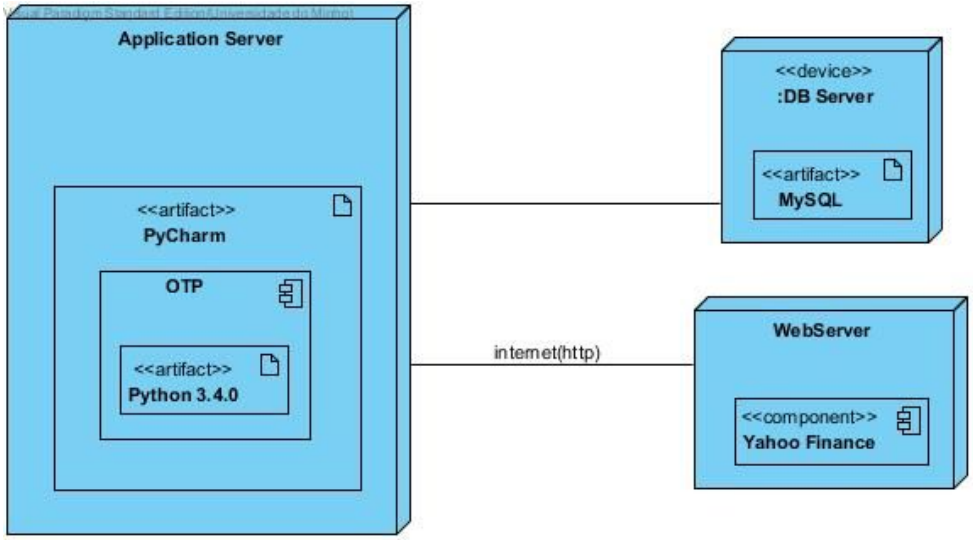


Fig. 2. Deployment Context Diagram

- Application Server: Local onde o programa irá ser utilizado. O utilizador deverá ter a ferramenta PyCharm, onde poderá executar a aplicação. Deverá ter também a versão Python 3.4.0 instalada, e fazer import do MySQL Connector;
- Web Server: A aplicação utiliza dados dos ativos, fornecidos pelo Yahoo Finance;
- DB Server: Os dados dos ativos, contratos e utilizadores estarão armazenados numa base de dados MySQL.

4. Estratégia de Solução

Numa plataforma online de trading é fundamental ter os valores dos ativos financeiros corretos. Para assegurar isso, a aplicação lê, constantemente, ficheiros .csv com valores (e outras informações relevantes) dos ativos e atualizados na base de dados. Quando estas alterações são feitas, os contratos em aberto de todos os utilizadores são verificados e atualizados. Deste modo, quando um utilizador faz login, os seus contratos estão atualizados.

5. Building Block View (Versão 1)

5. 1. Whitebox OTP

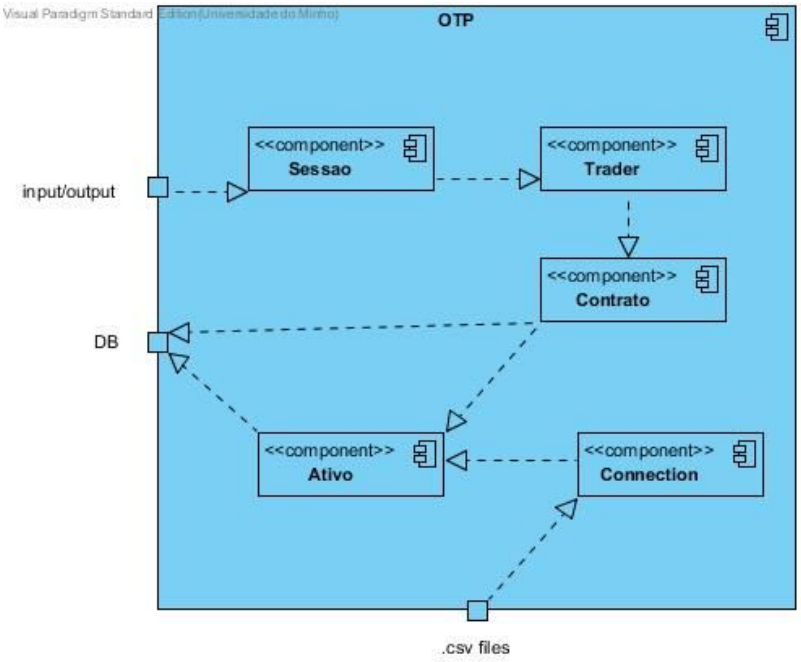


Fig. 3. Whitebox OTP

Tab. 7. Classes

Classe	**Descrição**
Sessao	Realiza comunicação entre cliente e a plataforma.
Trader	Responsável pelos métodos relativos a funcionalidades.
Contrato	Responsável pelos métodos relativos aos contratos.
Ativo	Responsável pelos métodos relativos a ativos financeiros.
Connection	Atualiza os valores dos ativos e dos contratos.

5. 1. 1. Blackbox Sessao

Esta classe é a responsável por iniciar a sessão do utilizador e fazer a comunicação entre utilizador e a plataforma.

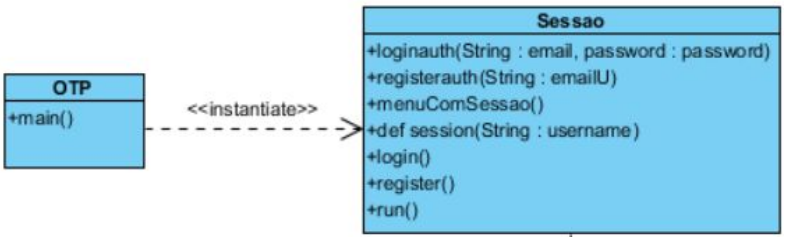


Fig. 4. Blackox Sessao

Tab. 8. Métodos

Método	Descrição
run	Fornece o menu inicial ao utilizador, com as opções de registar ou iniciar sessão.
login	Responsável pelo login do utilizador.
register	Responsável pelo registo do utilizador.
loginauth	Verifica se o email e a password do utilizador existem e correspondem existem na base de dados.
registerauth	Verifica se o email é válido, isto é, ainda não está na base de dados.
session	Guarda os dados da pessoa que iniciou sessão.
menuComSessao	Mostra as funcionalidades que o utilizador logado pode executar.

5. 1. 2. Blackbox Trader

Esta classe é responsável pelas funções que o trader pode realizar na aplicação. É instanciada pela classe Sessao.

Fig. 5. Blackbox Trader

5. 1. 1. Blackbox Contrato

Esta classe é responsável por todos os métodos relacionadas com contratos. é instanciada pela classe Trader.

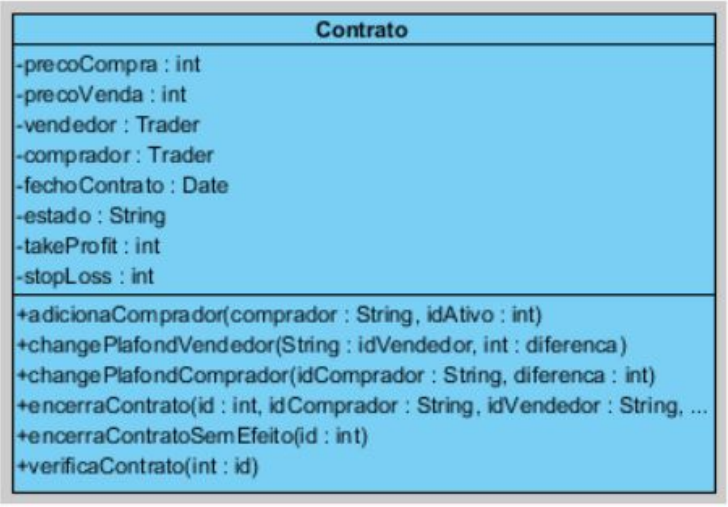


Fig. 6. Blackbox Contrato

Tab. 9. Métodos

Método	Descrição
adicionaComprador	Quando um comprador aceita um contrato, este método atualiza os dados do contrato, adicionando quem o aceitou e quantos contratos fez.
changePlafondVendedor	Quando um contrato é encerrado, este método atualiza o plafond do vendedor com o que ele ganhou/perdeu.
changePlafondComprador	Quando um contrato é encerrado, este método atualiza o plafond do comprador com o que ele ganhou/perdeu.
encerrarContrato	Fecha um contrato e atualiza o estado do mesmo.
encerrarContratoSemEfeito	Encerra um contrato que ninguém aceitou. Contrato sem efeito.
verificaContrato	Verifica se o contrato está em condições de ser encerrado. Este método é várias vezes chamado pela classe Connection.

5. 1. 1. Blackbox Ativo

Esta será a classe responsável pelos ativos. O seu único método é o atualizaAtivo, chamado pela classe Connection. Este método é responsável por alterar o valor de uma ativo em específico, na base de dados.

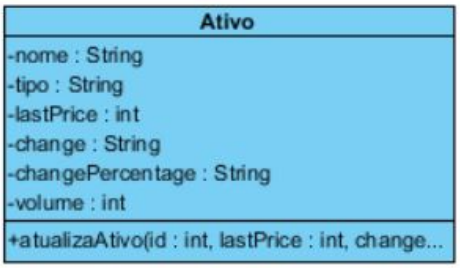


Fig. 7. Blackbox Ativo

5. 1. 1. Blackbox Connection

Esta classe estabelece conexão com a base de dados, e, lendo ficheiros .csv com os dados dos ativos, atualiza os valores na base de dados. Para além disso, lê os contratos abertos e verifica se os deve fechar. Em caso afirmativo instancia a classe Contrato, que irá fazer a gestão necessária.

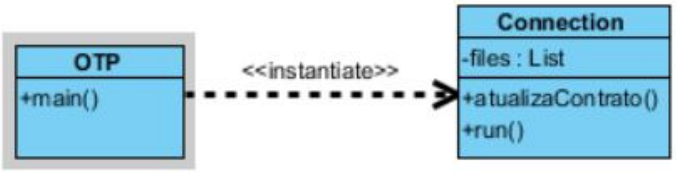


Fig. 8. Blackbox Connection

6. Runtime View

6. 1. Diagrama de Classes

- Na versão 1 da plataforma:

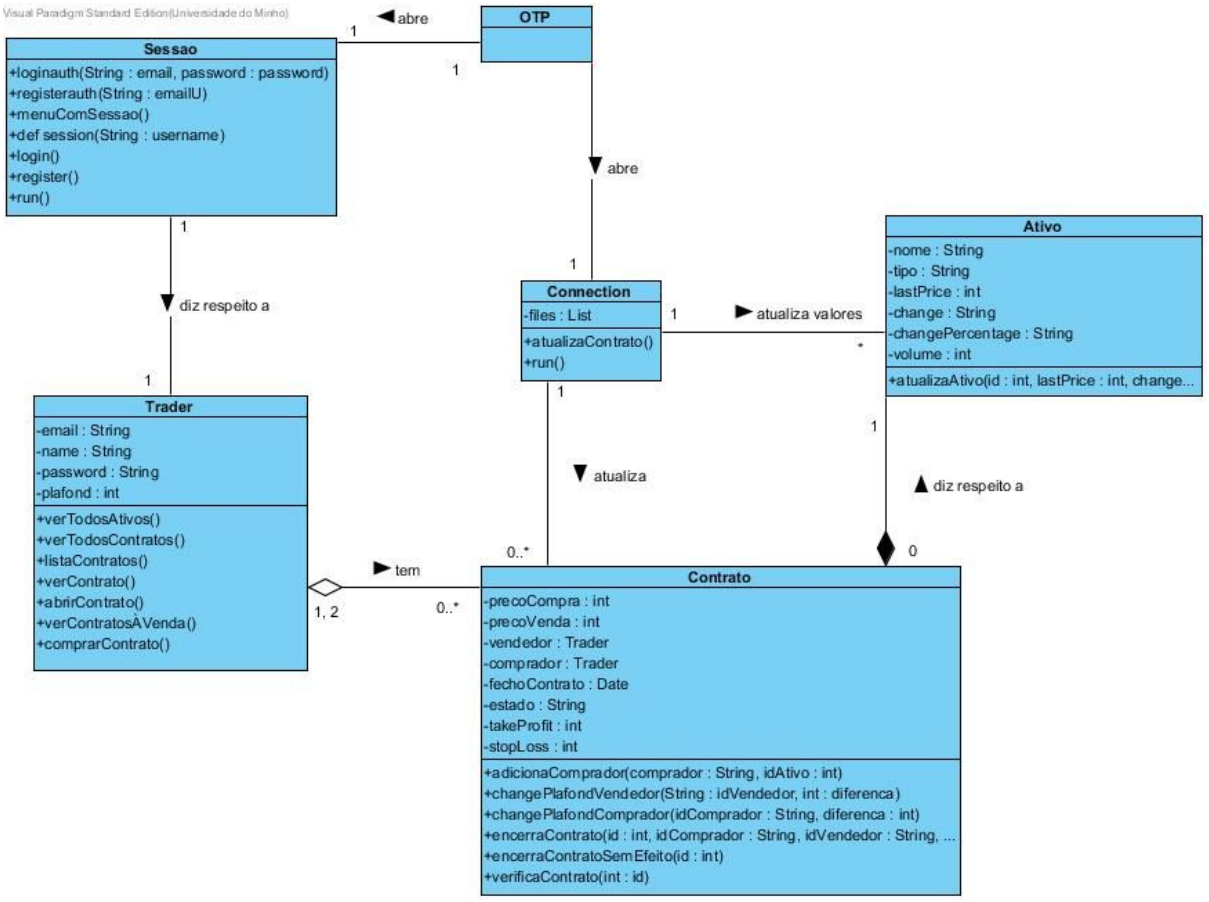


Fig. 9. Diagrama de Classes na versão 1

- Na versão 2 da plataforma:

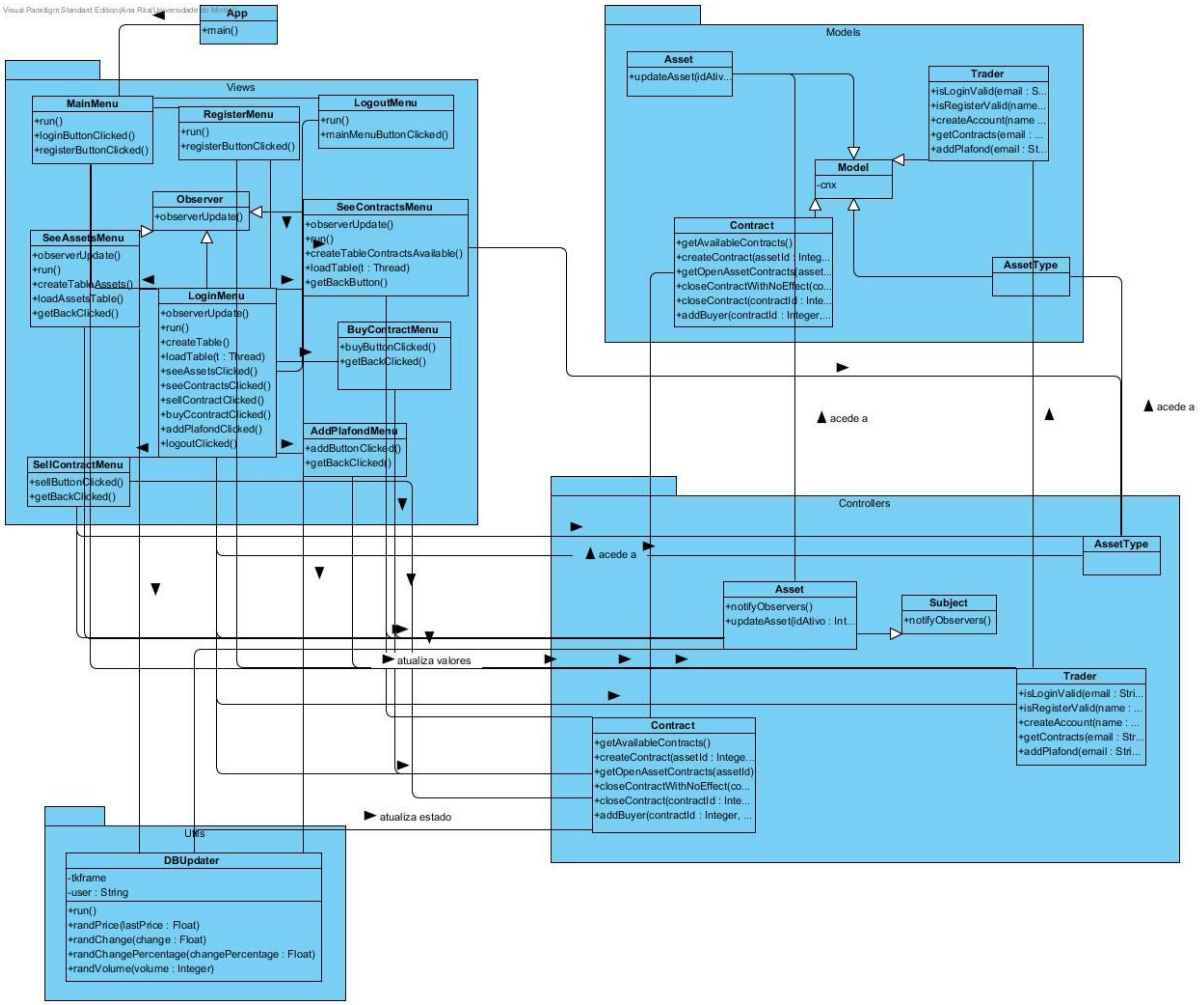


Fig. 10 Diagrama de Classes da versão 2

Como é um diagrama grande e confuso, decidi dividir em partes para uma melhor percepção:

- Dentro do *package Models:*

A classe *Model* é uma super classe, com o atributo cnx (conexão à base de dados). Todas as restantes classes deste package são uma subclasse de Model e utilizam a conexão herdada. Isto permite evitar estar a criar conexões para cada classe.

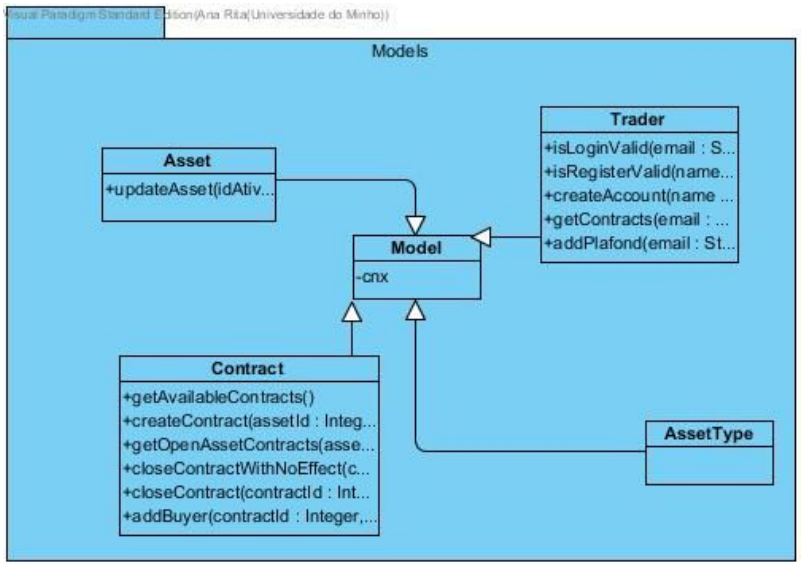


Fig. 11. Diagrama de classes do package Models

- Dentro do *package Controllers*:

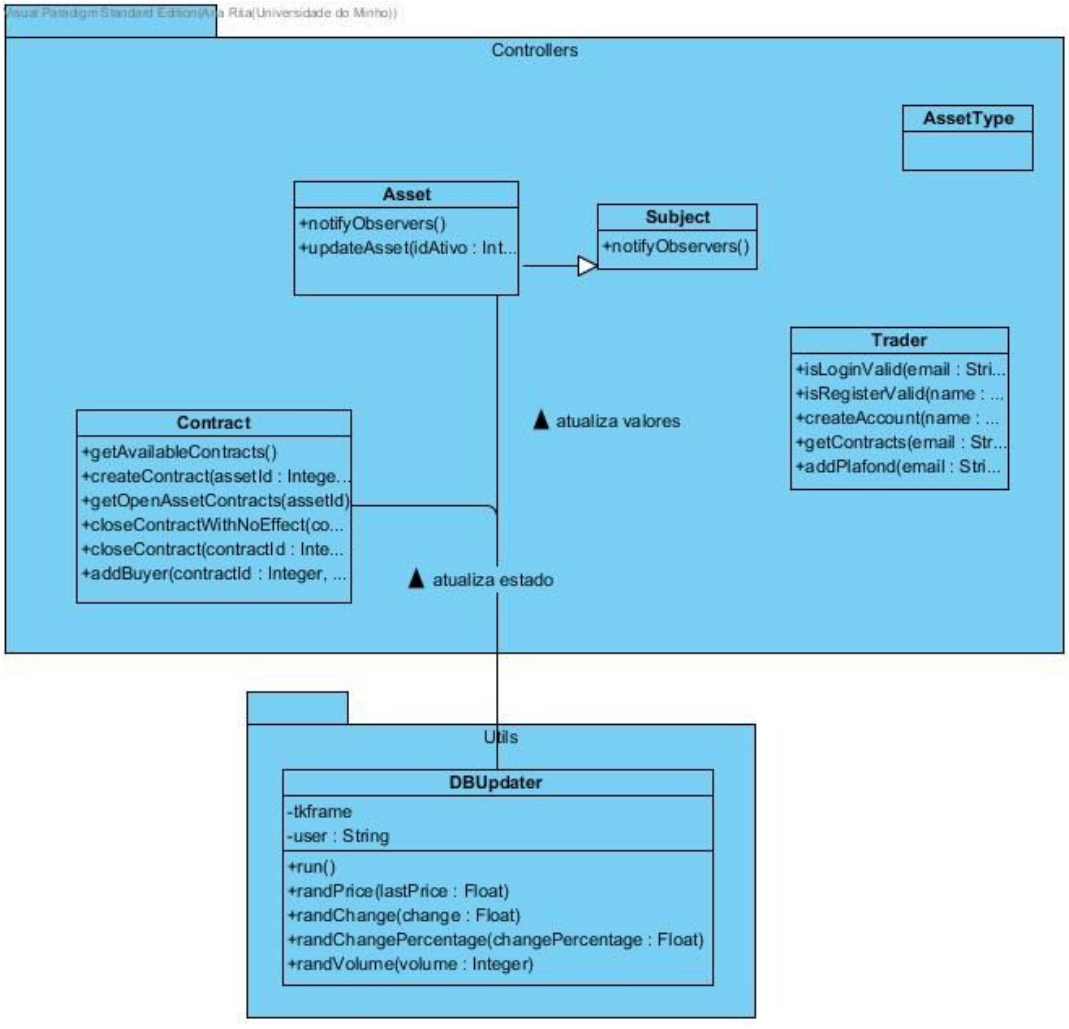


Fig. 12. Diagrama de classes do package Controllers

- Dentro do *package Views*:

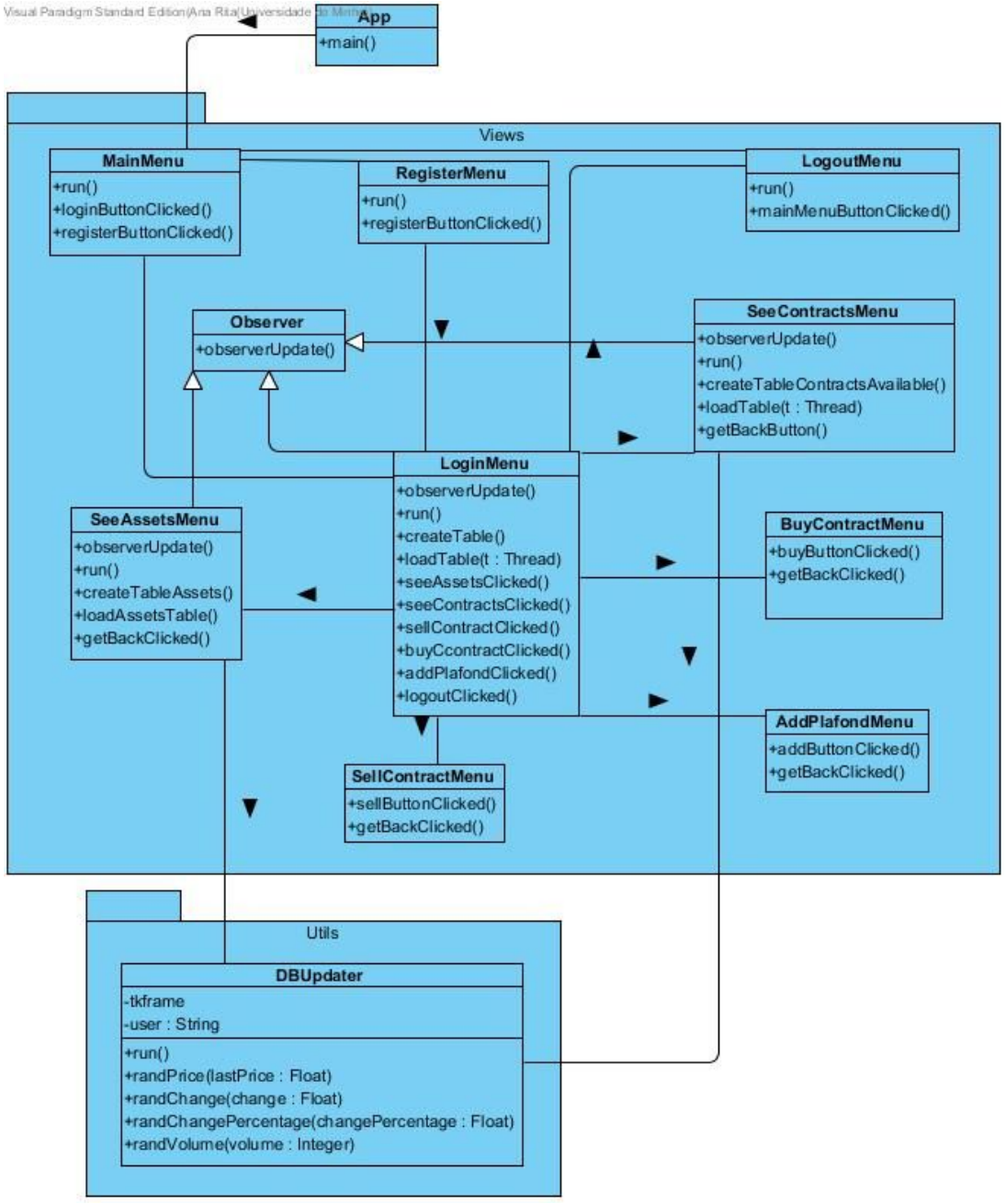


Fig. 13. Diagrama de classes do package Views

- *Controllers-Models*:

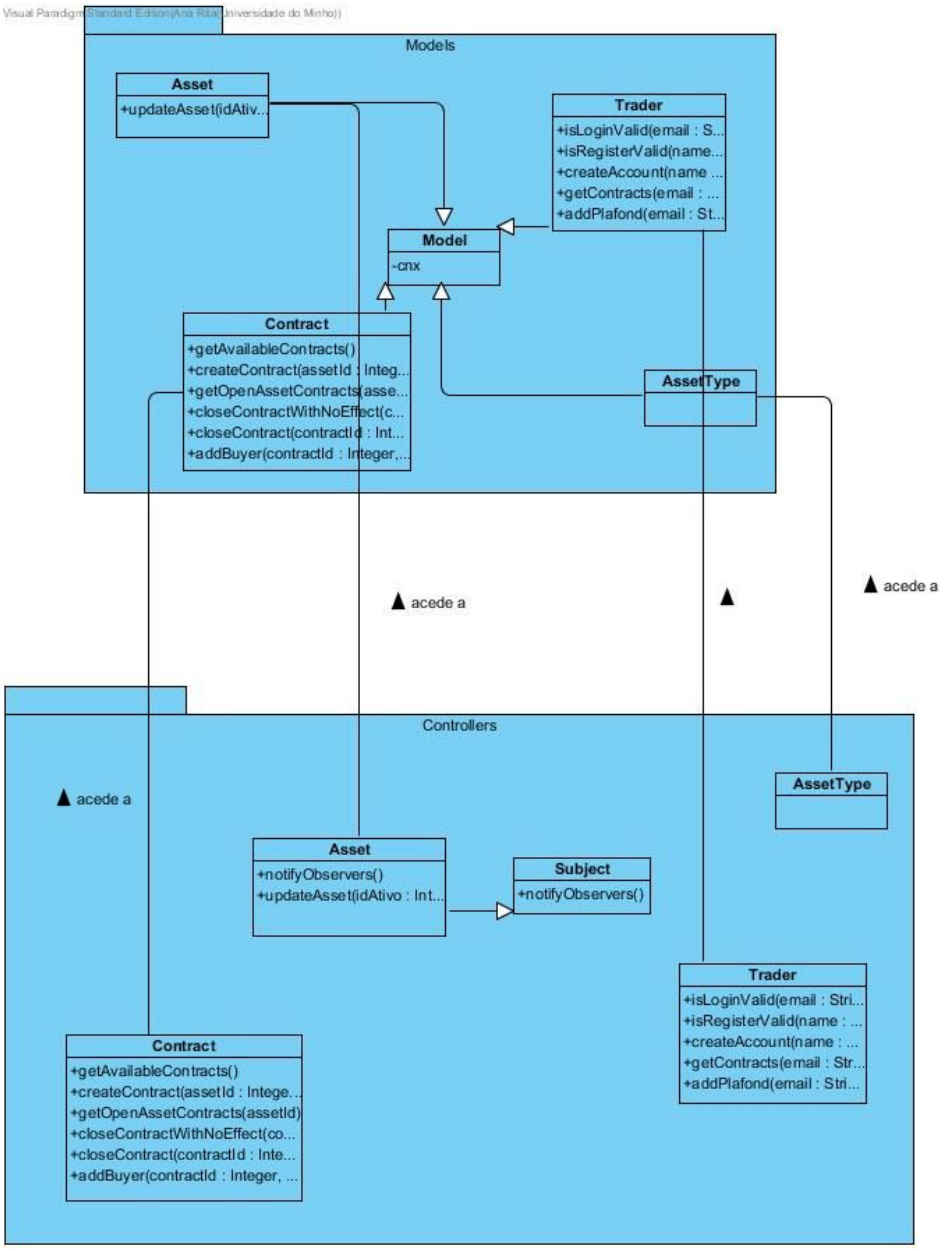


Fig. 14. Diagrama de classes Controllers-Models

- *Views-Controllers*:

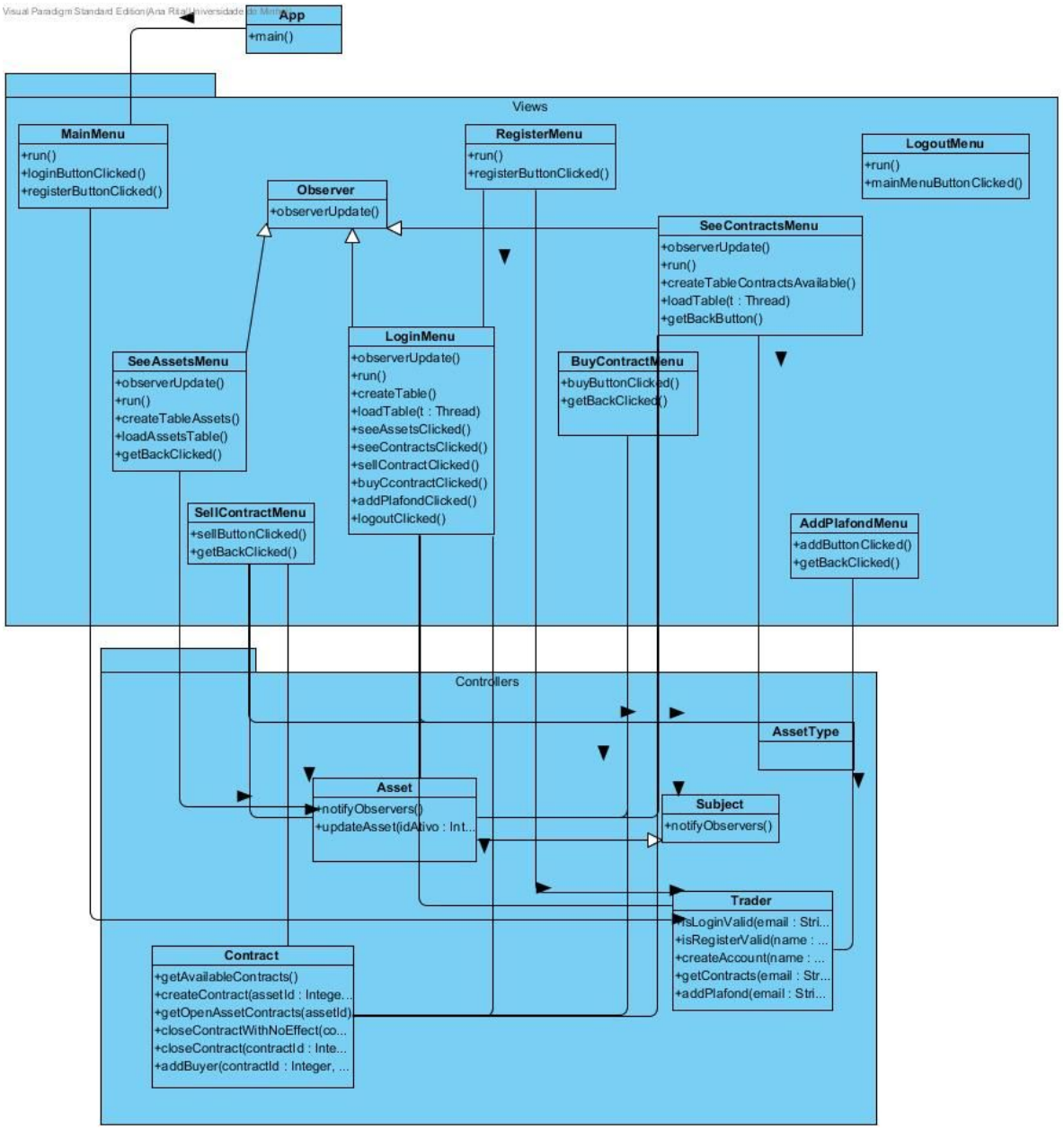


Fig. 15. Diagrama de Classes Views-Controllers

- A classe DBUpdater será responsável por gerar os valores para os ativos, atualizar a base de dados e verificar/encerrar contratos.

6. 2. Diagrama de Use Cases

- Na versão 1 da plataforma:

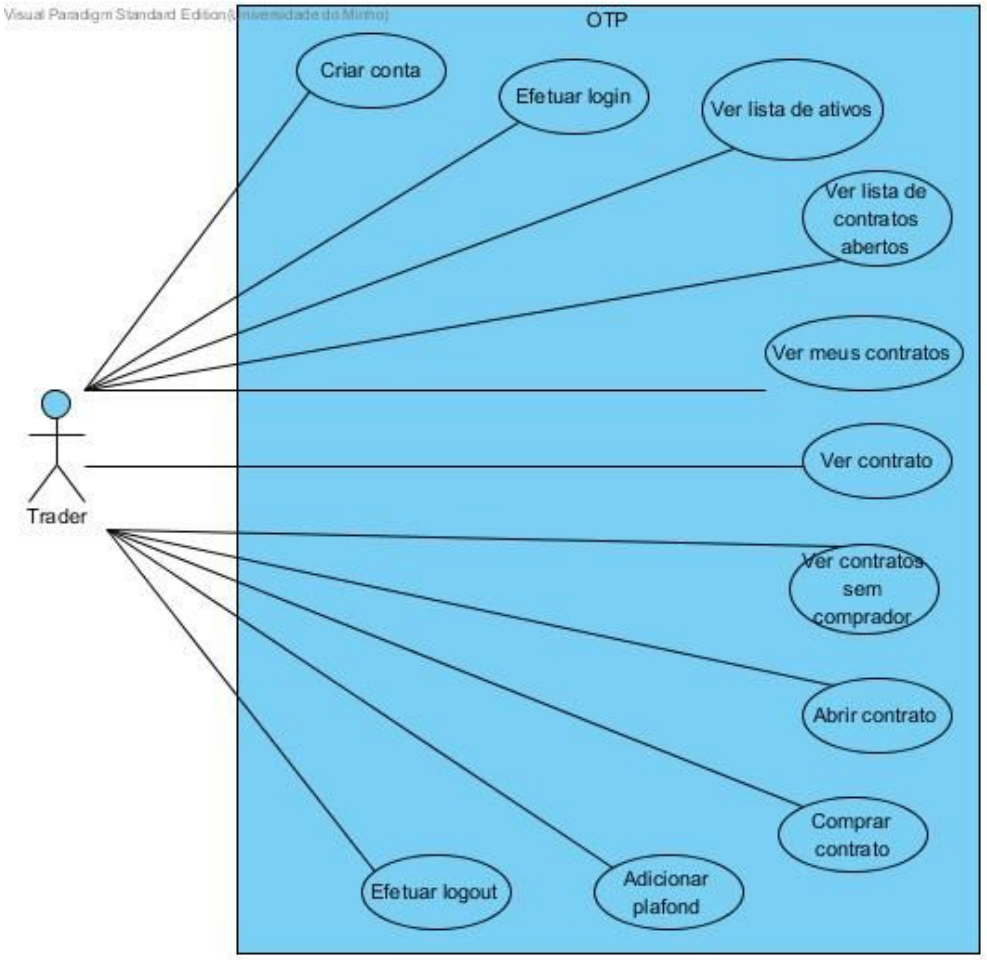


Fig. 16. Diagrama de Use Cases na versão 1

- Na versão 2 da plataforma:

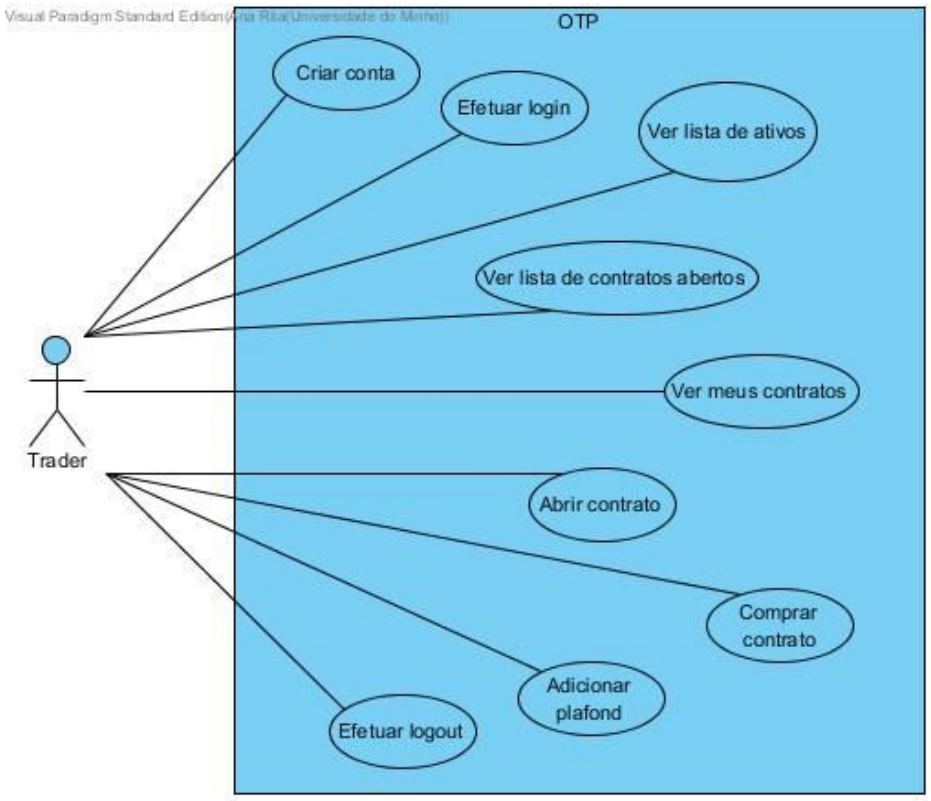


Fig. 17. Diagrama de Use Cases da Versão 2

6. 2. 1. Diagramas de Sequência de Use Case

6. 2. 1. 1. Ver contrato

Esta funcionalidade, que permite ver os detalhes de um determinado contrato, existe apenas na versão 1 da aplicação. Na versão 2 o utilizador consegue ver a lista completa de contratos disponíveis, com os respetivos detalhes.

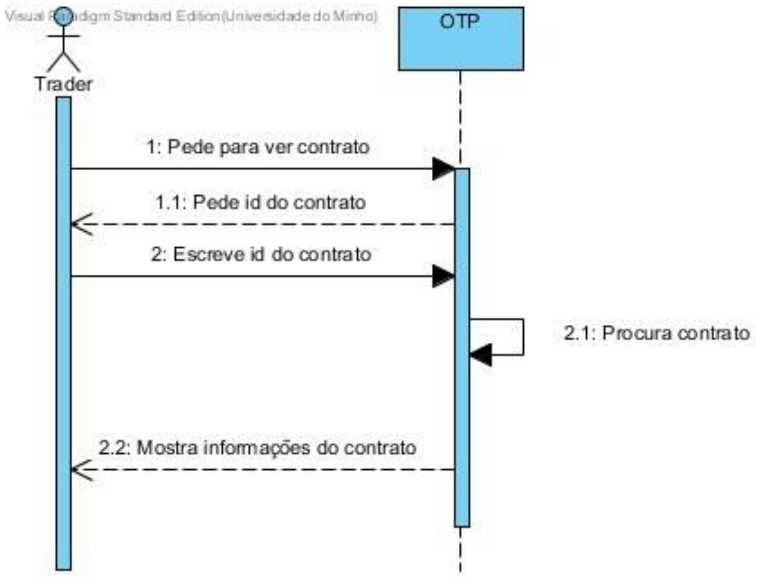


Fig. 18. Diagrama de Sequência de Ver Contrato

6. 2. 1. 2. Criar conta

- Na versão 1 da plataforma:

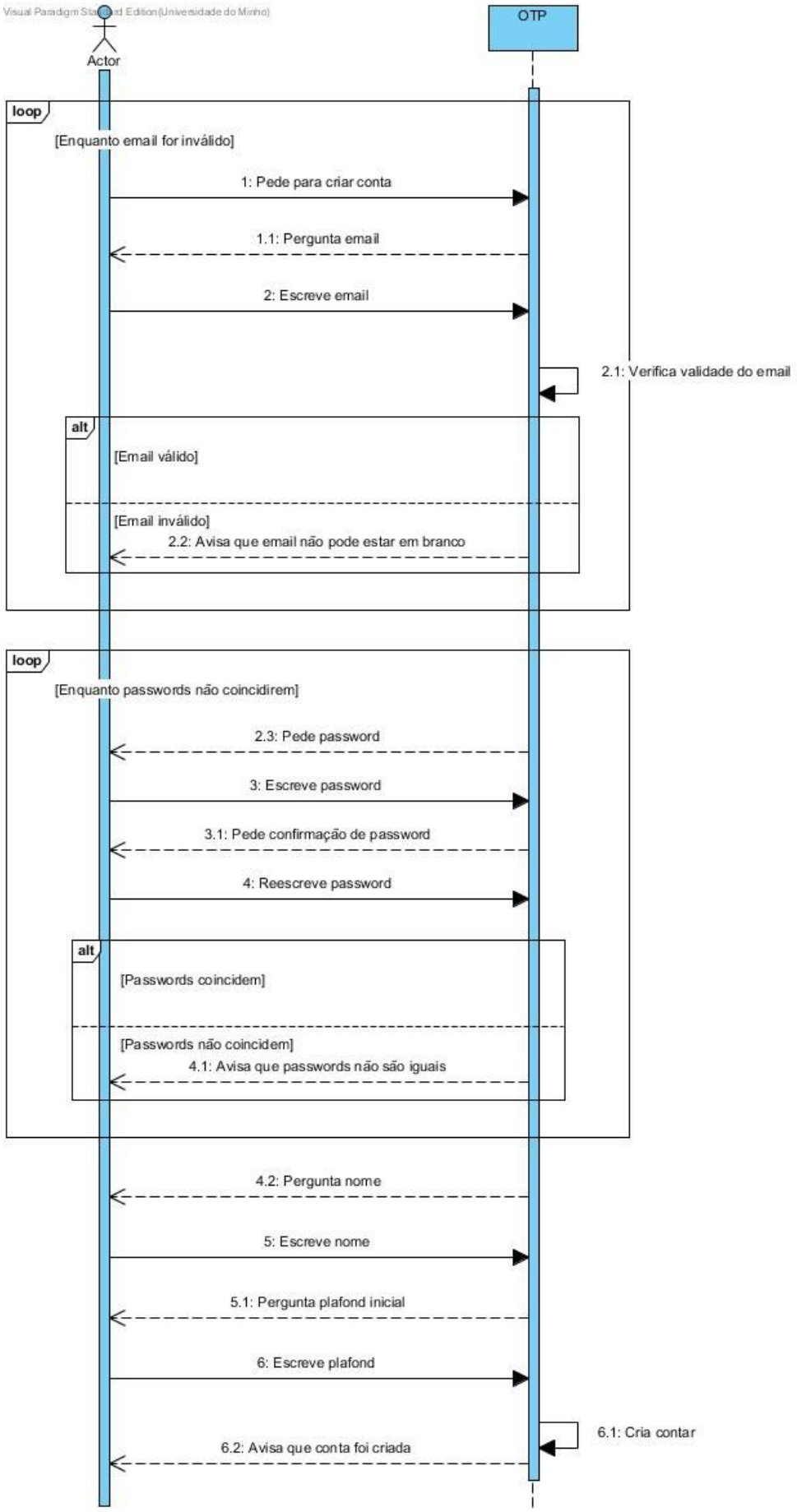

Visual Paradigm Standard Edition(Universidade do Minho)
Actor
OTP
loop
[Enquanto email for inválido]
1: Pede para criar conta
1.1: Pergunta email
2: Escreve email
2.1: Verifica validade do email
alt
[Email válido]
[Email inválido]
2.2: Avisa que email não pode estar em branco
loop
[Enquanto passwords não coincidirem]
2.3: Pede password
3: Escreve password
3.1: Pede confirmação de password
4: Reescreve password
alt
[Passwords coincidem]
[Passwords não coincidem]
4.1: Avisa que passwords não são iguais
4.2: Pergunta nome
5: Escreve nome
5.1: Pergunta plafond inicial
6: Escreve plafond
6.1: Cria contar
6.2: Avisa que conta foi criada

Fig. 19. Diagrama de Sequência de Criar Conta

- Na versão 2 da plataforma:

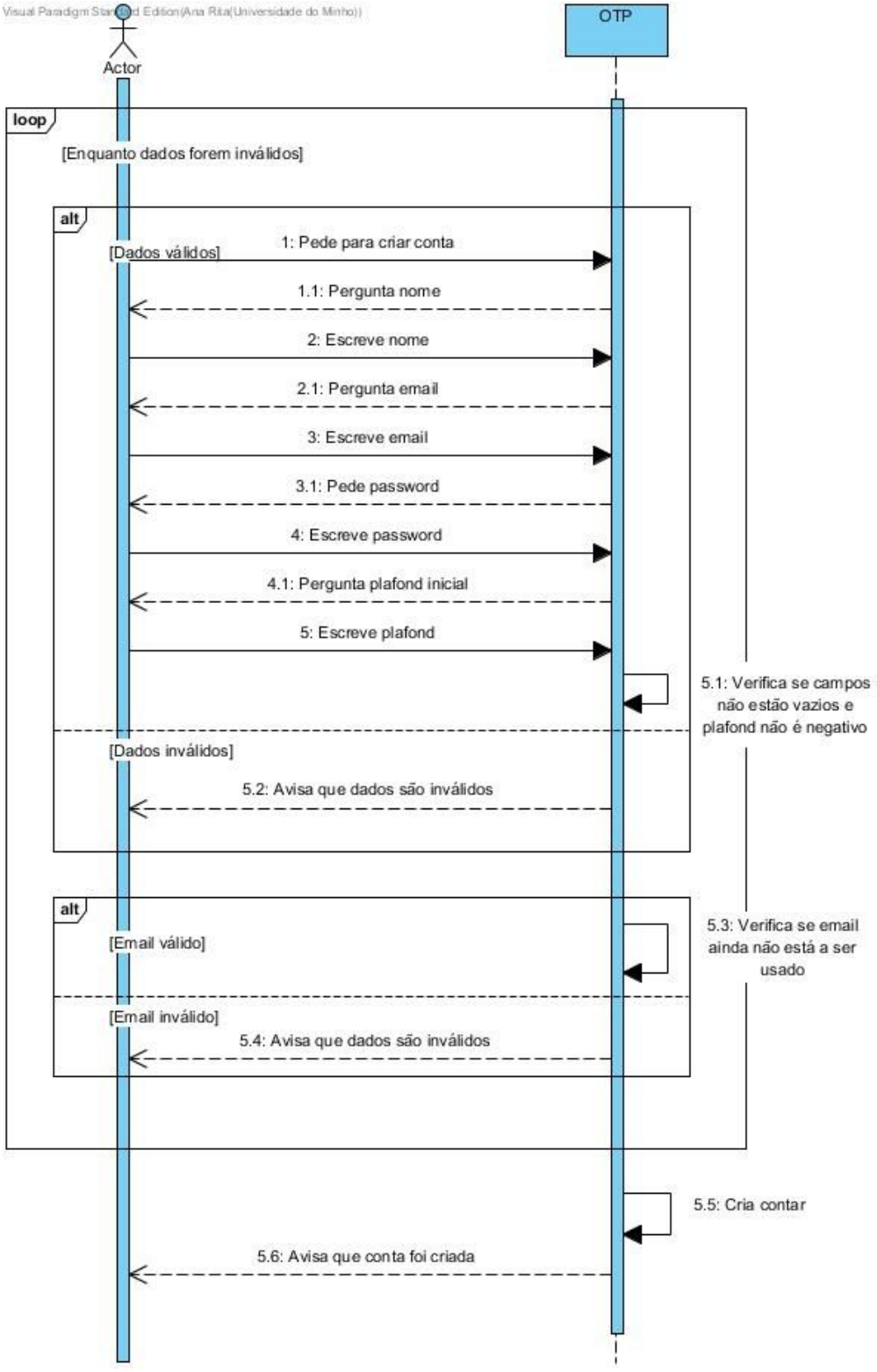


Fig. 20. Diagrama de Sequência de Criar Conta na Versão 2

6. 3. Diagramas de Sequência de Sistema

6. 3. 1. Comprar contrato

- Na versão 1 da plataforma:

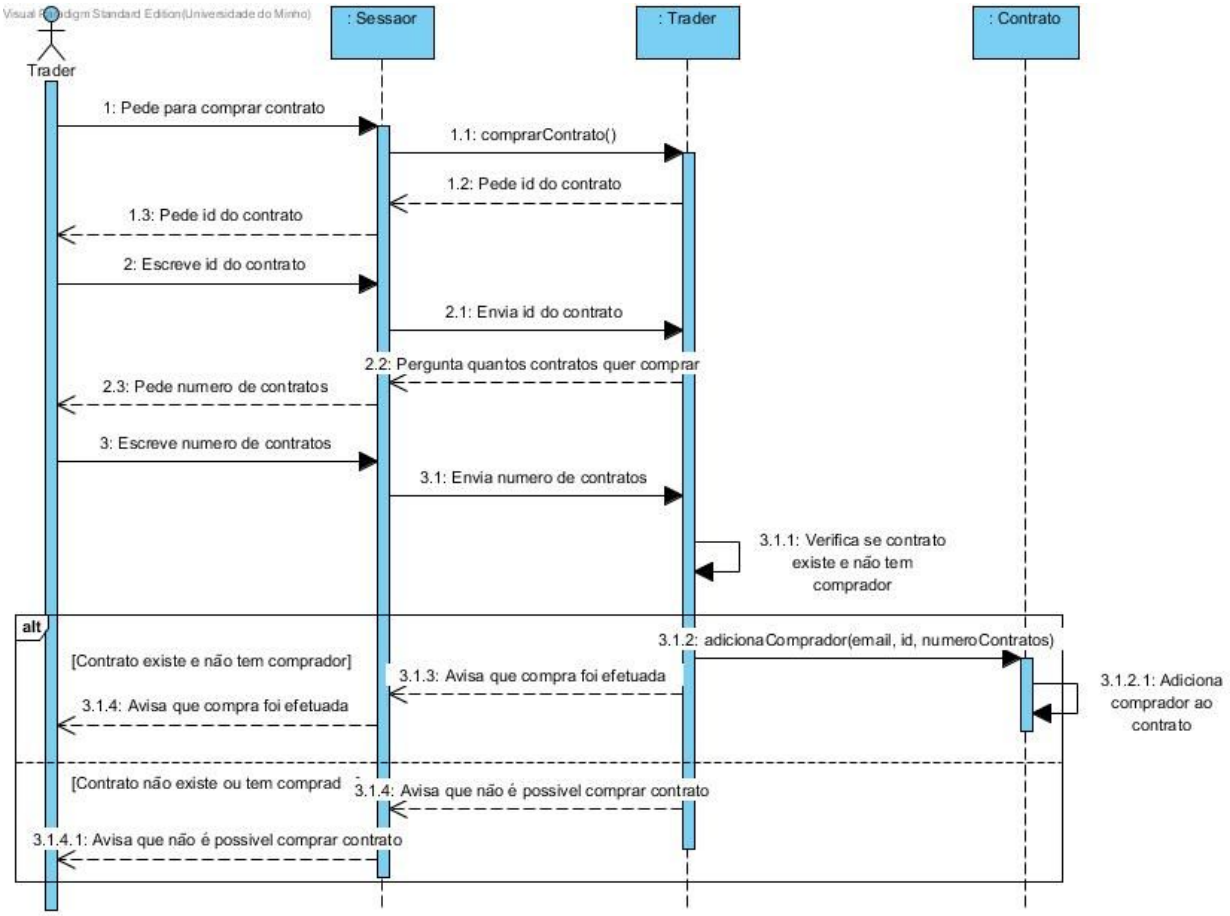


Fig. 21. Diagrama de Sequência de Comprar Contrato na Versão 1

- Na versão 2 da plataforma:

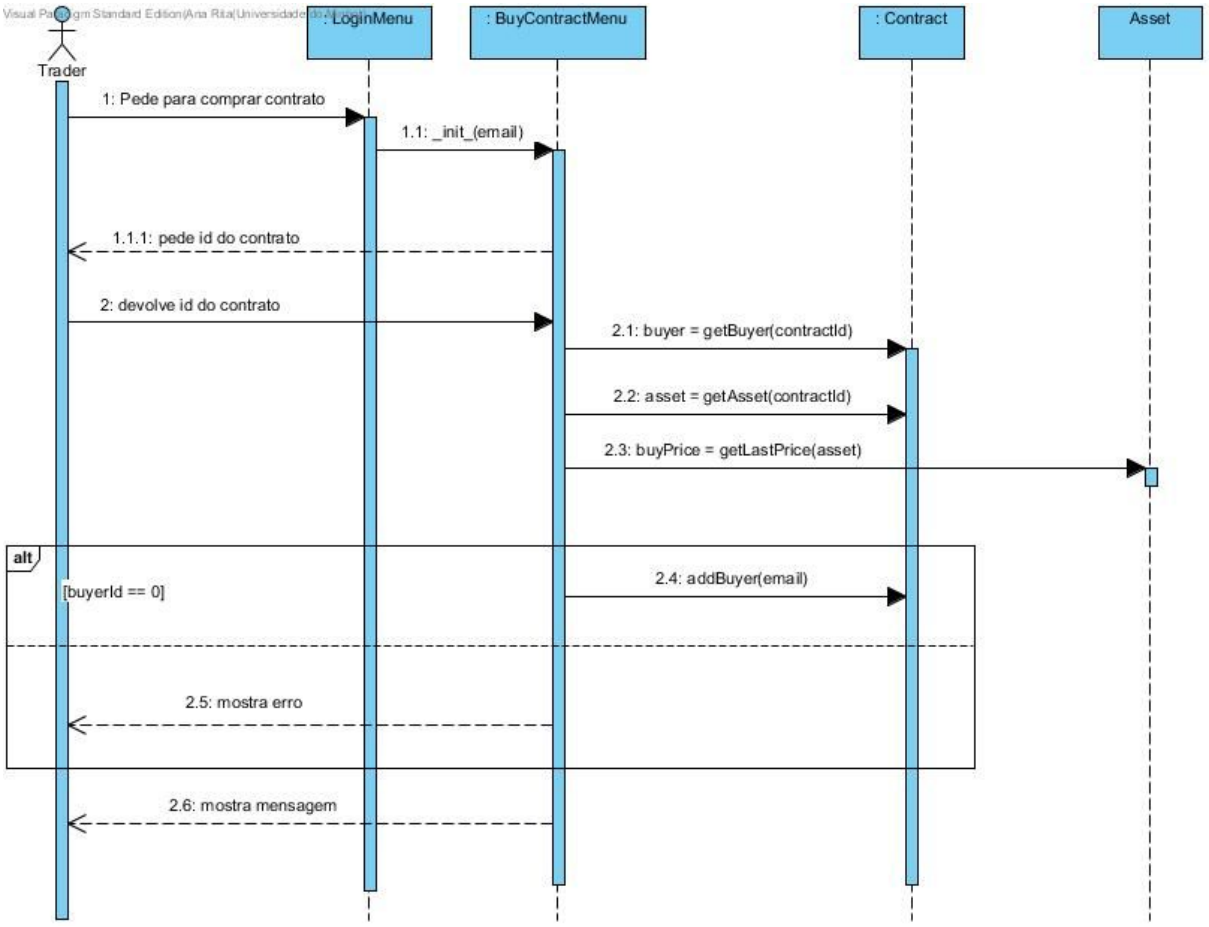


Fig. 22. Diagrama de Sequência de Comprar Contrato na Versão 2

6. 3. 2. Ver lista de ativos

- Na versão 1 da plataforma:

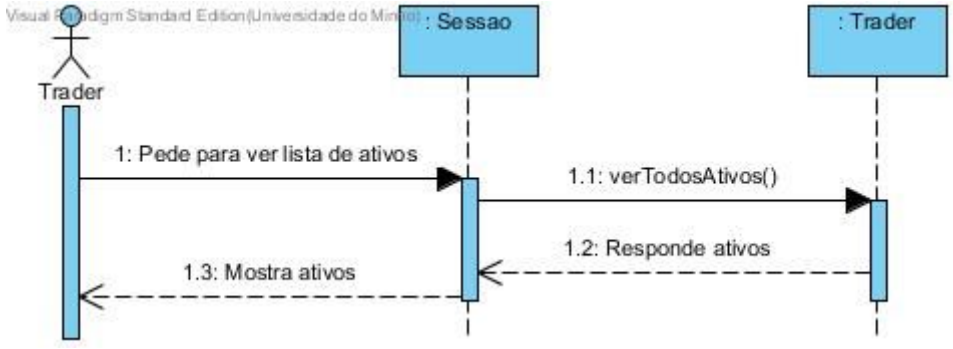


Fig. 23. Diagrama de Sequência de Ver Lista de Ativos na Versão 1

- Na versão 2 da plataforma:

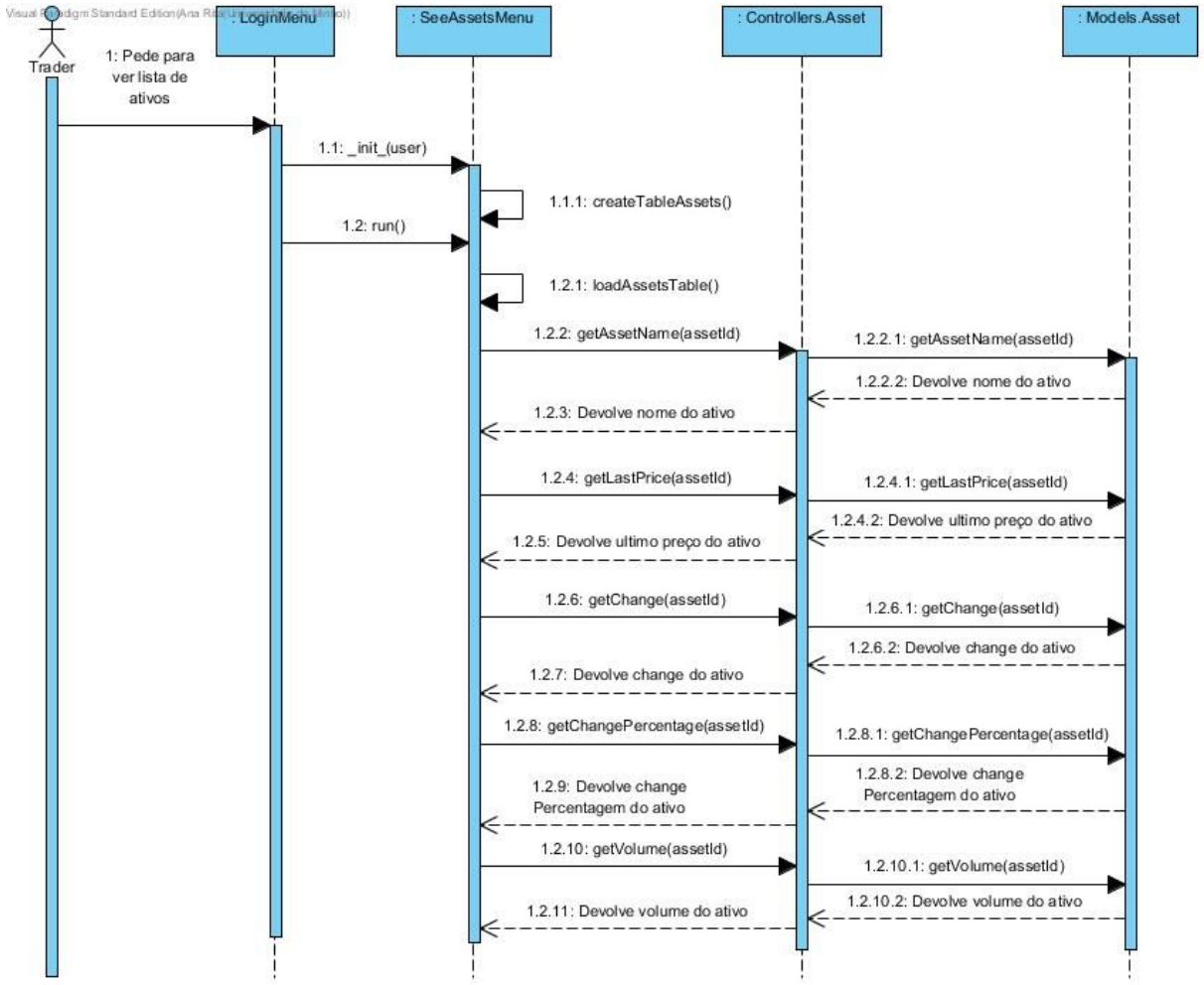


Fig. 24. Diagrama de Sequência de Ver Lista de Ativos na Versão 2

7. Visão de Implementação

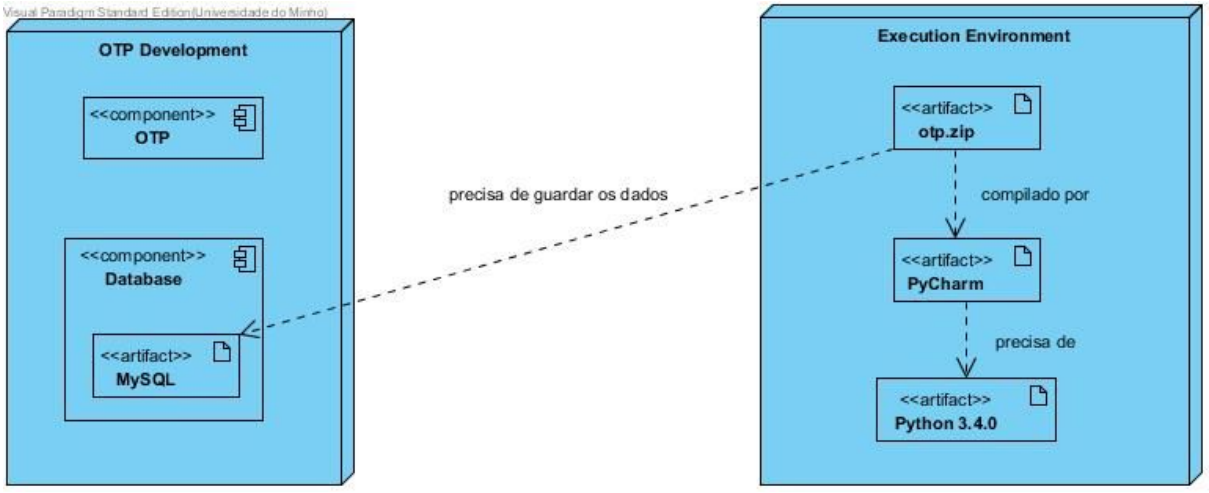


Fig. 25. Deployment Diagram

Node/Artifact	Descrição
OTP Development	Onde a OTP foi desenvolvida. Computador com Python 3.4.0 e onde a IDE PyCharm foi usada para compilar e correr o programa.
Execution Environment	Onde o utilizador irá correr a plataforma.
otp.zip	Onde estão todos os ficheiros necessários para a compilação do programa.
PyCharm	IDE sugerida para compilar e correr a plataforma.

7. 1. Base de Dados

É necessário que o utilizador tenha uma base de dados MySQL, já com um schema criado. O schema utilizado encontra-se em anexo.

A base de dados deverá também ter já alguns dados, que seguem também em anexo.

NOTA: Poderá ser necessário alterar os dados de conexão ao MySQL no ficheiro Model.py.

8. Conceitos

8. 1. Modelo de Domínio

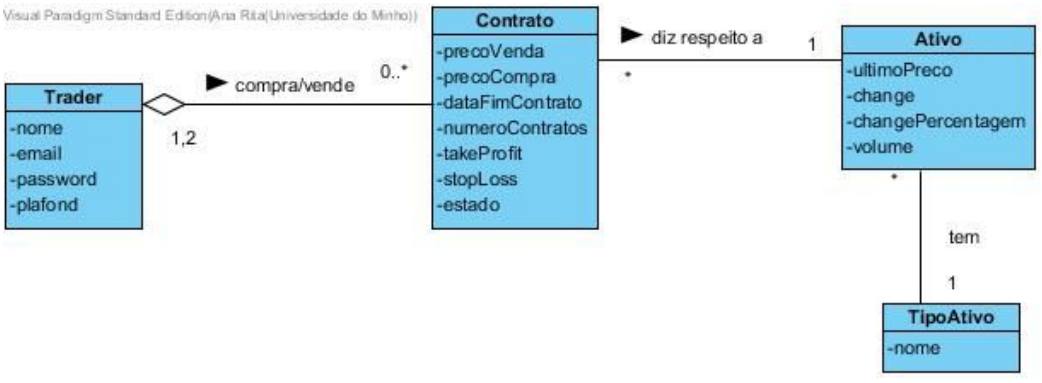


Fig. 26. Modelo de Domínio

8. 2. User Interface

A versão 1 da plataforma não tem User Interface, pelo que terá de ser corrida numa shell.

Na versão 2, foi implementada User Interface, recorrendo a um pacote de Python chamado Tkinter.

8. 3. Internacionalização

A língua suportada é Inglês.

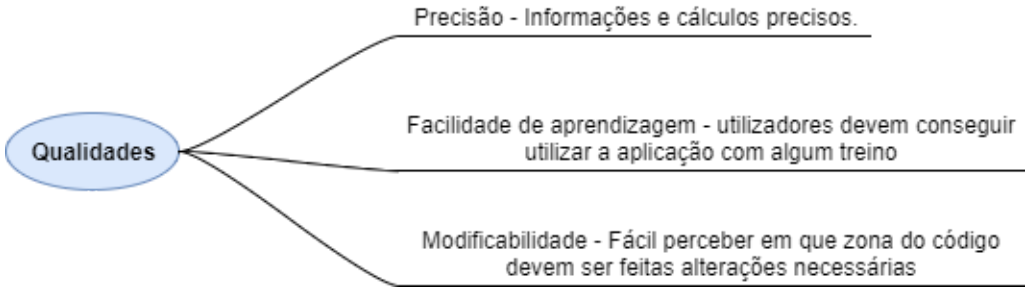
9. Decisões do design

A principal decisão neste projeto foi sobre a extração dos dados do Yahoo Finance. Após vária pesquisa a única possibilidade descoberta foi a extração de dados a partir de ficheiros .csv. Para cada ativo seria necessário um url diferente para o download dos dados, e por isso foram selecionados alguns ativos. Os ativos selecionados foram: Ouro, Prata, Crude, Google, Apple, Nvidia, Alibaba e IBM.

Numa segunda versão do projeto, a API ficou inutilizável, pelo que os valores dos ativos passaram a ser simulados com valores aleatórios.

10. Cenários de qualidade

10. 1. Quality Tree



11. Riscos Técnicos

A plataforma não foi testada por outros utilizadores e por isso desconhece-se potenciais riscos. Há a possibilidade de, por perdas de atualização de dados, alguns contratos sejam fechados com valores ligeiramente diferentes, prejudicando investimentos de grande valor. No entanto, estas correções seriam feitas num projeto com maior duração.

12. Mockups da Versão 1

12. 1. Página Inicial

Fig. 27. Mockup de Página Inicial

12. 2. Iniciar sessão

Fig. 28. Mockup de Iniciar Sessão

12. 3. Menu

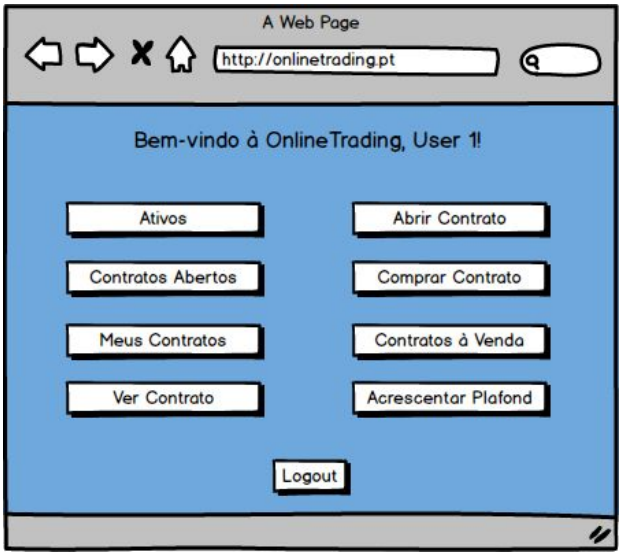


Fig. 29. Mockup de Menu

12. 4. Ativos

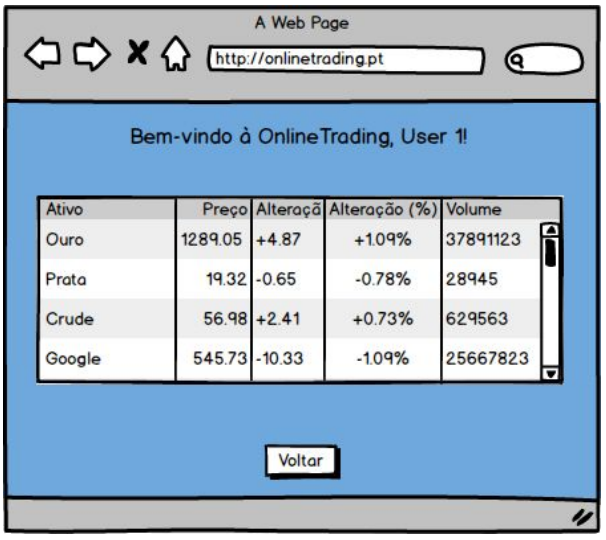


Fig. 30. Mockup de Ver Lista de Ativos

12. 5. Abrir Contrato

Fig. 31. Mockup de Abrir Contrato

13. Comparação entre Versão 1 e Versão 2

13. 1. Arquitetura da Plataforma

Numa primeira versão do trabalho, não é identificável nenhuma arquitetura, devido ao número reduzido de classes e à sua desorganização. Por esse mesmo

motivo, para a segunda versão do trabalho, decidi recomeçar a plataforma, implementando a arquitetura MVC. Para além disso, a versão anterior do trabalho não tinha interface gráfica, o que não permitia ao utilizador ver os valores a serem alterados em tempo real. Decidi então, nesta segunda versão, acrescentar interface gráfica. Deste modo, a parte respeitante à interface gráfica ficou no *package Views*, a parte respeitante a aceder à Base de Dados ficou no *package Models*, e no *package Controllers* ficaram as classes responsáveis por mediarem a comunicação entre *Views* e *Models*.

13. 2. Padrão de Design

Na segunda versão do trabalho foi implementado o padrão *observer*, onde o *subject* é o ativo e os *observers* são as *views*. Como *python* não tem interfaces, as classes *subject* e *observer* foram criadas como sendo *abstract*. Este padrão, quando os valores dos ativos são alterados, atualiza os respetivos valores na *view* em que o utilizador se encontra, caso seja uma *view* com tabelas de valores (contratos que o utilizador tem, todos os ativos e contratos disponíveis).

13. 3. Implementação de Novo Requisito

O requisito não foi implementado na aplicação, mas ponderei sobre as alterações que seriam necessárias:

- Acrescentar uma tabela na base de dados para registar os ativos que cada utilizador pretende marcar, assim como o valor em que pretende ser notificado;
- Na janela inicial, acrescentar a opção de seguir um ativo;
- Acrescentar um novo padrão *observer*, em que quando o valor de determinado ativo se altera, na classe DBUpdater, é feita uma pesquisa na base de dados, através da classe Models.Asset, para verificar se alguém está interessado nesse ativo e no seu novo valor; Em caso afirmativo, se o utilizador atual é um dos interessados, cujo email seria passado como argumento para a classe DBUpdater, a classe Controllers.Asset enviaria uma mensagem para a view onde o utilizador se encontrasse;
- Nas *views* seria acrescentado um método Update, que mostrasse a mensagem ao utilizador;

Módulos que seriam alterados:

- Base de Dados - nova tabela;
- Classe DBUpdater - chamaria método do Models.Asset;
- Models.Asset - criação de método que verifica se o utilizador está interessado nesse ativo e no seu novo valor;
- Controllers.Asset - criação de método que notifica as *views*;

- Views - criação de método Update, que envia uma mensagem ao utilizador. seria chamado no Controllers.Asset;

Seria possível reaproveitar o padrão observer já implementado, que notifica a *view* sempre que os valores dos ativos se alteram. Bastaria acrescentar a mensagem, caso a pessoa siga aquele ativo e valor.

14. Glossário

Termo	**Descrição**
MySQL	Sistema open source de gestão de bases de dados relacionais.
Yahoo Finance	Propriedade de media que faz parte da rede Yahoo!. Ele fornece notícias, dados e comentários financeiros, incluindo cotações de ações, comunicados de imprensa, relatórios financeiros e conteúdo original. Ele também oferece algumas ferramentas online para gerenciamento de finanças pessoais.
Python	Linguagem de programação de alto nível de propósito geral.
PyCharm	IDE usada especificamente para Python.
MVC	Model-View-Controller - Padrão arquitetônico

15. Anexos

1. Schema da Base de Dados

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
```

```
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema otp_db
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema otp_db
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `otp_db` DEFAULT CHARACTER SET utf8 ;
USE `otp_db` ;

-- -----------------------------------------------------
-- Table `otp_db`.`tipo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `otp_db`.`tipo` (
  `idTipo` INT(11) NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idTipo`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `otp_db`.`ativo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `otp_db`.`ativo` (
  `idAtivo` INT(11) NOT NULL,
  `idTipo` INT(11) NOT NULL,
  `ultimoPreco` FLOAT NOT NULL,
  `change` FLOAT NULL DEFAULT NULL,
  `changePercentagem` FLOAT NULL DEFAULT NULL,
  `volume` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`idAtivo`),
  INDEX `idTipo_idx` (`idTipo` ASC),
  CONSTRAINT `idTipo`
    FOREIGN KEY (`idTipo`)
    REFERENCES `otp_db`.`tipo` (`idTipo`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `otp_db`.`trader`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `otp_db`.`trader` (
  `email` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `plafond` FLOAT NOT NULL,
  PRIMARY KEY (`email`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `otp_db`.`contrato`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `otp_db`.`contrato` (
  `idContrato` INT(11) NOT NULL AUTO_INCREMENT,
  `idAtivo` INT(11) NOT NULL,
  `idVendedor` VARCHAR(45) NOT NULL,
  `idComprador` VARCHAR(45) NULL DEFAULT NULL,
  `precoVenda` INT(11) NOT NULL,
  `precoCompra` INT(11) NULL DEFAULT NULL,
  `dataFechoContrato` DATE NOT NULL,
  `takeProfit` INT(11) NOT NULL,
  `stopLoss` INT(11) NOT NULL,
  `estado` VARCHAR(45) NOT NULL,
  `numeroContratos` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`idContrato`),
  INDEX `idAtivo_idx` (`idAtivo` ASC),
  INDEX `idVendedor_idx` (`idVendedor` ASC),
  INDEX `idComprador_idx` (`idComprador` ASC),
  CONSTRAINT `idAtivo`
    FOREIGN KEY (`idAtivo`)
    REFERENCES `otp_db`.`ativo` (`idAtivo`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT `idComprador`
    FOREIGN KEY (`idComprador`)
    REFERENCES `otp_db`.`trader` (`email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `idVendedor`
    FOREIGN KEY (`idVendedor`)
    REFERENCES `otp_db`.`trader` (`email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 10
DEFAULT CHARACTER SET = utf8;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

2. Dados já na DB

```
INSERT INTO trader VALUES ('email1@otp.pt', 'pass1', 'nome1', '2000'),
('email2@otp.pt', 'pass2', 'nome2', '1000'),
('email3@otp.pt', 'pass3', 'nome3', '500'),
('email4@otp.pt', 'pass4', 'nome4', '10000'),
('email5@otp.pt', 'pass5', 'nome5', '4500');

INSERT INTO tipo VALUES (1, 'Gold'), (2, 'Silver'), (3, 'Crude Oil'), (4, 'Google'), (5,
'Apple'), (6, 'Nvidia'), (7, 'Alibaba'), (8, 'IBM');

INSERT INTO ativo VALUES(1, 1, 1297.00, '-7.60', '-0.58', 262723), (2, 2, 17.25,
'-0.16', '-0.92', 75763), (3, 3, 51.85, '+0.40', '+0.78', 492353), (4, 4, 991.92, '+2.24',
'+0.23', 836220), (5, 5, 159.88, '+2.89', '+1.84', 23898054), (6, 6, 197.93, '+3.34',
'+1.72', 14279079), (7, 7, 179.56, '+1.11', '+0.62', 12728649), (8, 8, 146.76, '-0.34',
'-0.23', 2581419);

INSERT INTO contrato (idAtivo, idVendedor, precoVenda, dataFechoContrato,
takeProfit, stopLoss, estado, numeroContratos)
```

```
VALUES(3, 'email2@otp.pt', 56.6, '2017-12-29', 55, 45, 'aberto', 10),
(5, 'email1@otp.pt', 159.76, '2017-12-29', 165, 155, 'aberto', 5),
(1, 'email3@otp.pt', 1287.3, '2017-12-29', 1350, 1200, 'aberto', 1),
(2, 'email4@otp.pt', 16.99, '2017-12-29', 21, 14, 'aberto', 100),
(8, 'email2@otp.pt', 151.87, '2017-12-29', 160, 140, 'aberto', 50),
(6, 'email5@otp.pt', 216.96, '2017-12-29', 230, 200, 'aberto', 23);

INSERT INTO contrato VALUES (1, 1, 'email1@otp.pt', 'email2@otp.pt', 1290,
NULL, '2017-12-01', 1310, 1270, 'aberto', 5),
(2, 3, 'email3@otp.pt', 'email2@otp.pt', 58.95, NULL, '2018-01-23', 65, 50, 'aberto',
20),
(3, 2, 'email4@otp.pt', 'email1@otp.pt', 16.99, NULL, '2018-01-23', 20, 10, 'aberto',
200),
(4, 4, 'email1@otp.pt', 'email5@otp.pt', 1040.61, NULL, '2018-01-23', 1100, 1000,
'aberto', 2),
(5, 7, 'email2@otp.pt', 'email5@otp.pt', 191.19, NULL, '2018-01-23', 210, 180,
'aberto', 25),
(6, 7, 'email3@otp.pt', 'email1@otp.pt', 191.19, NULL, '2018-01-23', 200, 185,
'aberto', 10),
(7, 8, 'email5@otp.pt', 'email4@otp.pt', 151.84, NULL, '2018-01-23', 170, 140,
'aberto', 15),
(8, 2, 'email4@otp.pt', 'email1@otp.pt', 16.99, NULL, '2018-01-23', 22, 12, 'aberto',
150);
```