**CryptoChallenge**

Cenários Propostos:

* Criação de um Website contendo um jogo relacionado com Crypto moedas;
* Desenvolver o Front-End e Backend e efetuar o deploy num servidor para ficar hosted na Web;
* Obtenção de data-series de um período aleatório (1 mês de valores) de uma Base de Dados dedicada a crypto moedas para garantir um jogo justo.

Objetivos do Jogo

* Criação de um grupo de jogadores para competir entre si;
* Atribuição de uma carteira inicial com fundos virtuais para transações;
* Simular um período temporal com as respetivas oscilações dos preços das moedas, sendo esse período uma proxy de um período de 1 mês real comprimido em apenas 3 a 5 minutos de jogo, permitindo a compra e venda das moedas em qualquer momento do jogo;
* Calcular o valor final da carteira dos jogadores, dando um ranking final e determinando o(s) vencedor(es). Mostrar um ranking final.

Arquitetura do Projeto

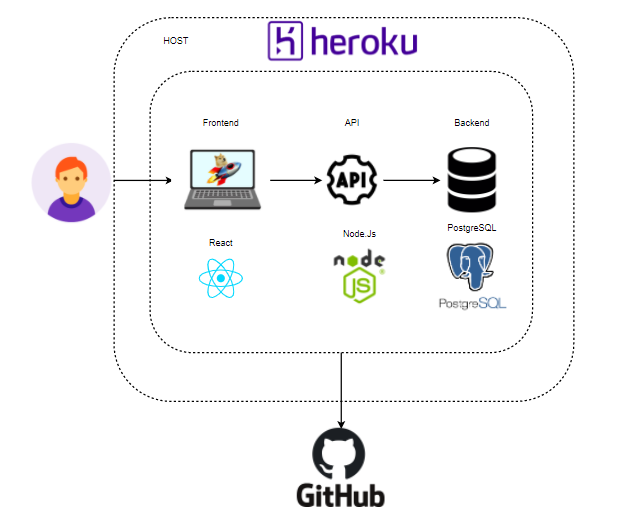


Figura 1- Arquitetura

|  |  |  |  |
| --- | --- | --- | --- |
| ENDPOINTS | | | |
| RESTFull URL | **HTTP Action** | **Noun** | **Business Operation** |
| /Users | GET | USERS | Get all users |
| ‘test(\'Test #1 - Listar os utilizadores\', () => {\r\n return request(app).get(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body).not.toHaveLength(0);\r\n expect(res.body[0]).toHaveProperty(\'firstName\');\r\n });\r\n});’ | | | |
| /Users/ | POST | USERS | InsertNewUser |
| ‘ test(\'Test #2 - Inserir utilizadores\', () => {\r\n return request(app).post(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({ firstName: \'Jo\u00E3o\', lastName: \'Manuel\', username: `${Date.now()}`, \r\n email: `${Date.now()}@gmail.com`, password: \'12345\' })\r\n .then((res) => {\r\n expect(res.status).toBe(201);\r\n expect(res.body.firstName).toBe(\'Jo\u00E3o\');\r\n expect(res.body).not.toHaveProperty(\'password\');\r\n });\r\n});’ | | | |
| ‘ test(\'Test #2.1 - Guardar a palavra passe encriptada\', async () => {\r\n const res = await request(app).post(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({ firstName: \'Jo\u00E3o\', lastName: \'Manuel\', username: `${Date.now()}`, \r\n email: `${Date.now()}@gmail.com`, password: \'12345\' })\r\n expect(res.status).toBe(201);\r\n\r\n const { id } = res.body;\r\n const userDB = await app.services.user.findOne({ id });\r\n expect(userDB.password).not.toBeUndefined();\r\n expect(userDB.password).not.toBe(\'12345\');\r\n});’ | | | |
| ‘describe(\' Test 2.2 - Valida\u00E7\u00E3o de criar um user\', () => {\r\n const testTemplate = (newData, errorMessage) => {\r\n return request(app).post(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({\r\n firstName: \'Jo\u00E3o\', lastName: \'Manuel\', username: `${Date.now()}`, \r\n email: `${Date.now()}@gmail.com`, password: \'12345\', ...newData,\r\n })\r\n .then((res) => {\r\n expect(res.status).toBe(400);\r\n expect(res.body.error).toBe(errorMessage);\r\n });\r\n };\r\n\r\n test(\'Test 2.2.1 - Sem First Name\', () => testTemplate({ firstName: null }, \'O First Name \u00E9 um atributo obrigat\u00F3rio\'));\r\n test(\'Test 2.2.2 - Sem Last Name\', () => testTemplate({ lastName: null }, \'O Last Name \u00E9 um atributo obrigat\u00F3rio\'));\r\n test(\'Test 2.2.3 - Sem Email\', () => testTemplate({ email: null }, \'O EMAIL \u00E9 um atributo obrigat\u00F3rio\'));\r\n test(\'Test 2.2.4 - Sem Username\', () => testTemplate({ username: null }, \'O USERNAME \u00E9 um atributo obrigat\u00F3rio\'));\r\n test(\'Test 2.2.5 - Sem Password\', () => testTemplate({ password: null }, \'A Password \u00E9 um atributo obrigat\u00F3rio\'));\r\n});’ | | | |
| ‘test('Test #2.3 - Inserir username Duplicado', () => {  return app.db('users')  .insert({ firstName: 'Zé', lastName: 'Manuel', username: 'UserDuplicated',  email: `${Date.now()}@gmail.com`, password: '12345'})  .then(() => request(app).post(MAIN\_ROUTE)  .set('authorization', `bearer ${user.token}`)  .send({firstName: 'Zé', lastName: 'Manuel', username: 'UserDuplicated',  email: `${Date.now()}@gmail.com`, password: '12345'}))  .then((res) => {  expect(res.status).toBe(400);  expect(res.body.error).toBe('Já existe uma conta com o username indicado');  });  });  test('Test #2.4 - Inserir email Duplicado', () => {  return app.db('users')  .insert({ firstName: 'Zé', lastName: 'Manuel', username: `${Date.now()}@gmail.com`,  email: email, password: '12345'})  .then(() => request(app).post(MAIN\_ROUTE)  .set('authorization', `bearer ${user.token}`)  .send({ firstName: 'Zé', lastName: 'Manuel', username: `${Date.now()}@gmail.com`,  email: email, password: '12345'}))  .then((res) => {  expect(res.status).toBe(400);  expect(res.body.error).toBe('Já existe uma conta com o email indicado');  });  });’ | | | |
| /Users/:id | GET | USERS | GetSpecificUser |
| ‘ test(\'Teste #3 - Listar um user por id\', () => {\r\n return app.db(\'users\')\r\n .insert({ firstName: \'Jo\u00E3o\', lastName: \'Manuel\', username: `${Date.now()}`, \r\n email: `${Date.now()}@gmail.com`, password: \'12345\' }, [\'id\'])\r\n .then((user) => request(app).get(`${MAIN\_ROUTE}\/${user[0].id}`)\r\n .set(\'authorization\', `bearer ${user.token}`))\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body[0].firstName).toBe(\'Jo\u00E3o\');\r\n expect(res.body[0]).not.toHaveProperty(\'password\');\r\n });\r\n});’ | | | |
| /Users/:id | DELETE | USERS | DeleteAccount |
| ‘test(\'Test #4 - Eliminar User\', () => {\r\n return app.db(\'user\')\r\n .insert({ firstName: \'Z\u00E9\', lastName: \'Manuel\', username: `${Date.now()}@gmail.com`, \r\n email: `${Date.now()}@gmail.com`, password: \'12345\'}, [\'id\'])\r\n .then((user) => request(app).delete(`${MAIN\_ROUTE}\/${user[0].id}`)\r\n .set(\'authorization\', `bearer ${user.token}`))\r\n .then((res) => {\r\n expect(res.status).toBe(204);\r\n });\r\n});’ | | | |
| /Users/:id | PUT | USERS | Update user information |
| ‘ test(\'Test #5 - Alterar info de utilizador por Id\', async () => {\r\n return app.db(MAIN\_ROUTE)\r\n .insert({ firstName: \'OldName\', lastName: \'Manuel\', username: `${Date.now()}`, \r\n email: `${Date.now()}@gmail.com`, password: \'12345\'}, [\'id\'])\r\n .then((acc) => request(app).put(`${MAIN\_ROUTE}\/${acc[0].id}`)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({ firstName: \'NewName\' }))\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body.firstName).toBe(\'NewName\');\r\n });\r\n});’ | | | |
| /Cryptos | GET | CRYPTO | GetAllCryptos |
| ‘ test(\'Test #6 - Listar todas as cryptos\', () => {\r\n return request(app).get(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body).not.toHaveLength(0);\r\n expect(res.body[0]).toHaveProperty(\'name\');\r\n });\r\n});’ | | | |
| /Cryptos/;<cryptoData> | POST | CRYPTO | Create new Crypto |
| ‘ test(\'Test #7 - Inserir crypto\', () => {\r\n return request(app).post(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({\r\n name: name,\r\n })\r\n .then((res) => {\r\n expect(res.status).toBe(201);\r\n expect(res.body.name).toBe(name);\r\n });\r\n});\r\n\r\ntest(\'Test #7.1 - Inserir nome de Crypto duplicado\', () => {\r\n return app.db(\'cryptos\')\r\n .insert({ name: \'Crypto Dup\' })\r\n .then(() => request(app).post(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({ name: \'Crypto Dup\' }))\r\n .then((res) => {\r\n expect(res.status).toBe(400);\r\n expect(res.body.error).toBe(\'Name duplicado na Bd\');\r\n });\r\n});’ | | | |
| /Cryptos/:id/ | PUT | CRYPTO | UpdateCryptoData |
| ‘ test(\'Test #8 - Alterar Crypto por ID\', () => {\r\n return app.db(\'cryptos\')\r\n .insert({ name: \'Crypto - Update\'}, [\'id\'])\r\n .then((crypto) => request(app).put(`${MAIN\_ROUTE}\/${crypto[0].id}`)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({ name: \'Crypto Updated\' }))\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body.name).toBe(\'Crypto Updated\');\r\n });\r\n});’ | | | |
| /Cryptos/:id/ | DELETE | CRYPTO | RemoveCrypto |
| ‘ test(\'Test #9 - Remover Crypto\', () => {\r\n return app.db(\'cryptos\')\r\n .insert({ name: \'Crypto to Remove\' }, [\'id\'])\r\n .then((crypto) => request(app).delete(`${MAIN\_ROUTE}\/${crypto[0].id}`)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({ name: \'Crypto Removed\' }))\r\n .then((res) => {\r\n expect(res.status).toBe(204);\r\n });\r\n});’ | | | |
| /Cryptos/:id/ | GET | CRYPTO | GetSepcificCrypto |
| test(\'Teste #10 - Listar uma Crypto por id\', () => {\r\n return app.db(\'cryptos\')\r\n .insert({ name: crypto}, [\'id\'])\r\n .then((crypto) => request(app).get(`${MAIN\_ROUTE}\/${crypto[0].id}`)\r\n .set(\'authorization\', `bearer ${user.token}`))\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body[0].name).toBe(crypto);\r\n expect(res.body).toHaveLength(1);\r\n });\r\n}); | | | |
| /auth/signin | POST | USER | Login |
| ‘ test(\'Teste #11.1 - Receber token ao autenticar\', () => {\r\n return app.services.user.save(\r\n { firstName: \'Pedro\', lastName: \'Auth\', email: email, username: username, password: \'12345\' },\r\n ).then(() => request(app).post(\'\/auth\/signin\')\r\n .send({ username: username, password: \'12345\' }))\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body).toHaveProperty(\'token\');\r\n });\r\n});\r\n\r\ntest(\'Teste #11.2 - Tentativa de autentica\u00E7ao errada\', () => {\r\n const wrongUsername = `${Date.now()}`;\r\n return app.services.user.save(\r\n { firstName: \'Pedro\', lastName: \'Auth\', email: `${Date.now()}@ipca.pt`, username: wrongUsername, password: \'12345\' },\r\n ).then(() => request(app).post(\'\/auth\/signin\')\r\n .send({ username: wrongUsername, password: \'67890\' }))\r\n .then((res) => {\r\n expect(res.status).toBe(400);\r\n expect(res.body.error).toBe(\'Autentica\u00E7\u00E3o Inv\u00E1lida!\');\r\n });\r\n});\r\n\r\ntest(\'Teste #11.3 - Tentativa de autentica\u00E7ao com utilizador errado\', () => {\r\n const invalidUsername = `${Date.now()}`;\r\n return request(app).post(\'\/auth\/signin\')\r\n .send({ username: invalidUsername, password: \'67890\' })\r\n .then((res) => {\r\n expect(res.status).toBe(400);\r\n expect(res.body.error).toBe(\'Autentica\u00E7\u00E3o Inv\u00E1lida! #2\');\r\n });\r\n});\r\n\r\ntest(\'Teste #11.4 - Aceder a rotas protegidas\', () => {\r\n return request(app).get(\'\/v1\/users\')\r\n .then((res) => {\r\n expect(res.status).toBe(401);\r\n });\r\n}); ’ | | | |
| /auth/signup | POST | USER | CreateuserAccount |
| ‘ test(\'Teste #12 - Criar utilizador\', () => {\r\n const newUsername = `${Date.now()}`;\r\n return request(app).post(\'\/auth\/signup\')\r\n .send({ firstName: \'Pedro\', lastName: \'Signup\', email: `${Date.now()}@ipca.pt`, username: newUsername, password: \'12345\' })\r\n .then((res) => {\r\n expect(res.status).toBe(201);\r\n expect(res.body.firstName).toBe(\'Pedro\');\r\n expect(res.body.lastName).toBe(\'Signup\');\r\n expect(res.body).toHaveProperty(\'email\');\r\n expect(res.body).not.toHaveProperty(\'password\');\r\n });\r\n});’ | | | |
| /games | Post | GAMES | CreateNewGame |
| ‘test(\'Teste #13 - Criar um novo jogo\', () => {\r\n return request(app).post(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({\r\n startDate: startDate, endDate: Date.now()\r\n })\r\n .then((res) => {\r\n expect(res.status).toBe(201);\r\n expect(res.body.startDate).toBe(startDate);\r\n });\r\n});’ | | | |
| /games/:id | Get | GAMES | GetSpecificGame |
| ‘ test(\'Teste #14 - Listar um jogo por id\', () => {\r\n return app.db(\'games\')\r\n .insert({ startDate: startDate, endDate: Date.now()}, [\'id\'])\r\n .then((game) => request(app).get(`${MAIN\_ROUTE}\/${game[0].id}`)\r\n .set(\'authorization\', `bearer ${user.token}`))\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body.startDate).toBe(startDate);\r\n expect(res.body).toHaveLength(1);\r\n });\r\n});’ | | | |
| /games/:id/timer | PUT | GAMES | StartTimer |
| ‘ test(\'Teste #15 - definir o inicio-fim do jogo\', () => {\r\n const newStartDate = Date.now();\r\n const newEndDate = Date.now();\r\n\r\n return app.db(\'games\')\r\n .insert({ startDate: startDate, endDate: Date.now()}, [\'id\'])\r\n .then((game) => request(app).put(`${MAIN\_ROUTE}\/${game[0].id}`)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({ startDate: newStartDate, endDate: newEndDate }))\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body.startDate).toBe(newStartDate);\r\n expect(res.body.endDate).toBe(newEndDate);\r\n expect(res.body).toHaveLength(1);\r\n });\r\n});’ | | | |
| /LeaderBoards/:gameId | Get | LEARDERBOARD | CalculateLeaderBoardForSpecificGame |
| ‘ test(\'Teste #16 - Obter o leaderboard de um determinado jogo\', () => {\r\n return app.db(\'games\_users\').insert(\r\n {users\_id: user.id, game\_id: testGame.id, cashBalance: 20000 }, [\'id\'],\r\n )\r\n .then((game\_user) => app.db(\'leaderboards\').insert(\r\n {game\_user\_id: game\_user.id , finalCashBalance: game\_user.cashBalance }\r\n )\r\n .then(() => request(app).get(`${MAIN\_ROUTE}\/${testGame.id}`)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body[0].game\_user\_id).toBe(game\_user.id);\r\n expect(res.body[0].finalCashBalance).toBe(20000);\r\n })));\r\n});’ | | | |
| /GameWallet/:gameiD/:userId | GET | WALLET | GetuserSpecificGameWallet |
| test(\'Test #17.1 - Obter a gameWallet de um utilizador\', () => {\r\n return request(app).get(`${MAIN\_ROUTE}\/${testGame.id}\/${user.id}`)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body).not.toHaveLength(0);\r\n expect(res.body[0]).toHaveProperty(\'crypto\_id\');\r\n expect(res.body[0].amount).tobe(5);\r\n expect(res.body[0].games\_users\_id).tobe(testGameUser.id);\r\n });\r\n});\r\n\r\ntest(\'Test #17.2 - Obter as coins de outros utilizadores\', () => {\r\n return app.db(\'\/v1\/users\')\r\n .insert({ firstName: \'Account\', lastName: \'Invalid\', email: `${Date.now()}@ipca.pt`, username: `${Date.now()}`, password: \'12345\'}, [\'id\'])\r\n .then((newUser) => request(app).get(`${MAIN\_ROUTE}\/${testGame.id}\/${newUser.id}`)\r\n .set(\'authorization\', `bearer ${user.token}`))\r\n .then((res) => {\r\n expect(res.status).toBe(403);\r\n expect(res.body.error).toBe(\'N\u00E3o tem acesso ao recurso solicitado\');\r\n });\r\n});\r\n\r\ntest(\'Teste #17.3 - Tentar um transa\u00E7ao de venda sem possuir a cripto em carteira, ou em quantidade suficiente\', () => {\r\n return app.db(\'gameWallet\').insert(\r\n { games\_users\_id: testGameAUserA, crypto\_id: testCrypto.id, amount: 1 }, [\'id\'],\r\n ).then((gamewallet) => request(app).put(`${MAIN\_ROUTE}\/${gamewallet[0].id}`)\r\n .set(\'authorization\', `bearer ${userA.token}`)\r\n .send({ amount: -5 })\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body.desc).toBe(\'Trans Updated\');\r\n }));\r\n \r\n const res = await app.services.gameWallet.save({ games\_users\_id: testGameAUserA, crypto\_id: testCrypto.id, amount: 1});\r\n testGameAUserC = { ...res3[0] };\r\n\r\n return request(app).put(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${userC.token}`)\r\n .send({\r\n games\_users\_id: testGameAUserA, crypto\_id: testCrypto.id, amount: 1 \r\n })\r\n .then((res) => {\r\n expect(res.status).toBe(401);\r\n expect(res.body.error).toBe(\'N\u00E3o tem saldo suficiente para a transa\u00E7\u00E3o\');\r\n });\r\n});’ | | | |
| /GameUser/<gameid, userId> | PUT | TRANSACTION | TryPurchaseWithNoCash |
| test(\'Teste #18 - Tentar uma transa\u00E7ao de compra sem saldo suficiente\', async () => {\r\n return app.db(\'games\_users\').insert(\r\n {users\_id: user.id, game\_id: testGame.id, cashBalance: 0 }, [\'id\'],\r\n ).then((game\_user) => request(app).put(`${MAIN\_ROUTE}\/${game\_user[0].id}`)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({ cost: cryptoCost })\r\n .then((res) => {\r\n expect(res.status).toBe(401);\r\n expect(res.body.error).toBe(\'N\u00E3o tem saldo suficiente para a transa\u00E7\u00E3o\');\r\n }));\r\n}); | | | |
| /Transactions/:gameId/:userId | GET | TRANSACTION | GetUserTransactionHistory |
| ‘ test(\'Teste #19.1 - Listar todas as transa\u00E7oes de utilizador\', () => {\r\n return app.db(\'transactions\').insert([\r\n { games\_users\_id: testGameAUserA.id, crypto\_id: testCrypto.id, date: new Date(), type: \'B\', amount: 5, crypto\_value: 100 },\r\n { games\_users\_id: testGameAUserA.id, crypto\_id: testCrypto.id, date: new Date(), type: \'S\', amount: -2, crypto\_value: 100 },\r\n { games\_users\_id: testGameAUserB.id, crypto\_id: testCrypto.id, date: new Date(), type: \'S\', amount: -5, crypto\_value: 100 },\r\n ]).then(() => request(app).get(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${userA.token}`)\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body).toHaveLength(2);\r\n expect(res.body[0].desc).toBe(\'B\');\r\n expect(res.body[0].amount).toBe(5);\r\n }));\r\n});\r\n\r\ntest(\'Teste #19.2 - Aceder a transa\u00E7oes de outro utilizador\', () => {\r\n return app.db(\'transactions\').insert(\r\n { games\_users\_id: testGameAUserB.id, crypto\_id: testCrypto.id, date: new Date(), type: \'B\', amount: 5, crypto\_value: 100 }, [\'id\'],\r\n ).then((trans) => request(app).get(`${MAIN\_ROUTE}\/${trans[0].id}`)\r\n .set(\'authorization\', `bearer ${userA.token}`)\r\n .then((res) => {\r\n expect(res.status).toBe(403);\r\n expect(res.body.error).toBe(\'N\u00E3o tem acesso ao recurso solicitado\');\r\n }));\r\n});’ | | | |
| /Transactions/<gameid, userId> | POST | TRANSACTION | AddNewGameTransaction |
| ‘ test(\'Teste #20.1 - Inserir Transa\u00E7\u00E3o de um utilizador\', () => {\r\n return request(app).post(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${user.token}`)\r\n .send({\r\n games\_users\_id: testGameAUserA.id, crypto\_id: testCrypto.id, date: new Date(), type: \'B\', amount: 5, crypto\_value: 100 \r\n })\r\n .then((res) => {\r\n expect(res.status).toBe(201);\r\n expect(res.body.games\_users\_id).toBe(testGameAUserA.id);\r\n expect(res.body.type).toBe(\'B\');\r\n });\r\n});\r\n\r\n\r\n\r\ntest(\'Teste #20.2 - Transa\u00E7oes de compra devem ser positivas\', () => {\r\n return request(app).post(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${userA.token}`)\r\n .send({\r\n games\_users\_id: testGameAUserA.id, crypto\_id: testCrypto.id, date: new Date(), type: \'B\', amount: 5, crypto\_value: 100\r\n })\r\n .then((res) => {\r\n expect(res.status).toBe(201);\r\n expect(res.body.games\_users\_id).toBe(testGameAUserA.id);\r\n expect(res.body.amount).toBe(\'5.00\');\r\n });\r\n});\r\ntest(\'Teste #20.3 -Transa\u00E7oes de venda devem ser negativas\', () => {\r\n return request(app).post(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${userA.token}`)\r\n .send({\r\n games\_users\_id: testGameAUserA.id, crypto\_id: testCrypto.id, date: new Date(), type: \'S\', amount: -2, crypto\_value: 100\r\n })\r\n .then((res) => {\r\n expect(res.status).toBe(201);\r\n expect(res.body.games\_users\_id).toBe(testGameAUserA.id);\r\n expect(res.body.amount).toBe(\'-2.00\');\r\n });\r\n});\r\n\r\n\/\/validar todos os campos\r\ndescribe(\' 20.4 Valida\u00E7\u00E3o de criar uma transa\u00E7\u00E3o\', () => {\r\n const testTemplate = (newData, errorMessage) => {\r\n return request(app).post(MAIN\_ROUTE)\r\n .set(\'authorization\', `bearer ${userA.token}`)\r\n .send({\r\n games\_users\_id: testGameAUserC.id, crypto\_id: testCrypto.id, date: new Date(), type: \'B\', amount: 5, crypto\_value: 100, ...newData,\r\n })\r\n .then((res) => {\r\n expect(res.status).toBe(400);\r\n expect(res.body.error).toBe(errorMessage);\r\n });\r\n };\r\n\r\n test(\'Test 20.4.1 - Sem Game\_User\_ID\', () => testTemplate({ games\_users\_id: null }, \'O GAME\_USER \u00E9 um atributo obrigat\u00F3rio\'));\r\n test(\'Test 20.4.2 - Sem Crypto\_ID\', () => testTemplate({ crypto\_id: null }, \'O Crypto\_ID \u00E9 um atributo obrigat\u00F3rio\'));\r\n test(\'Test 20.4.3 - Sem Data\', () => testTemplate({ date: null }, \'A DATA \u00E9 um atributo obrigat\u00F3rio\'));\r\n test(\'Test 20.4.4 - Sem Tipo\', () => testTemplate({ type: null }, \'O TIPO \u00E9 um atributo obrigat\u00F3rio\'));\r\n test(\'Test 20.4.5 - Com Tipo errado\', () => testTemplate({ type: \'P\' }, \'O TIPO tem um valor inv\u00E1lido\'));\r\n test(\'Test 20.4.6 - Sem Amount\', () => testTemplate({ amount: null }, \'O AMOUNT \u00E9 um atributo obrigat\u00F3rio\'));\r\n test(\'Test 20.4.7 - Sem o valor da crypto\', () => testTemplate({ crypto\_value: null }, \'O VALOR DA CRYPTO \u00E9 um atributo obrigat\u00F3rio\'));\r\n}); ‘ | | | |
| /Transactions/:game/:userId/:transId | GET | TRANSACTION | GetSpecificTransaciton |
| ‘ test(\'Teste #21 - Listar uma transa\u00E7ao\', () => {\r\n return app.db(\'transactions\').insert(\r\n { games\_users\_id: testGameAUserA.id, crypto\_id: testCrypto.id, date: new Date(), type: \'B\', amount: 5, crypto\_value: 100 }, [\'id\'],\r\n ).then((trans) => request(app).get(`${MAIN\_ROUTE}\/${trans[0].id}`)\r\n .set(\'authorization\', `bearer ${userA.token}`)\r\n .then((res) => {\r\n expect(res.status).toBe(200);\r\n expect(res.body.id).toBe(trans[0].id);\r\n expect(res.body.games\_users\_id).toBe(testGameAUserA.id);\r\n }));\r\n});’ | | | |

Mockup (protótipo)

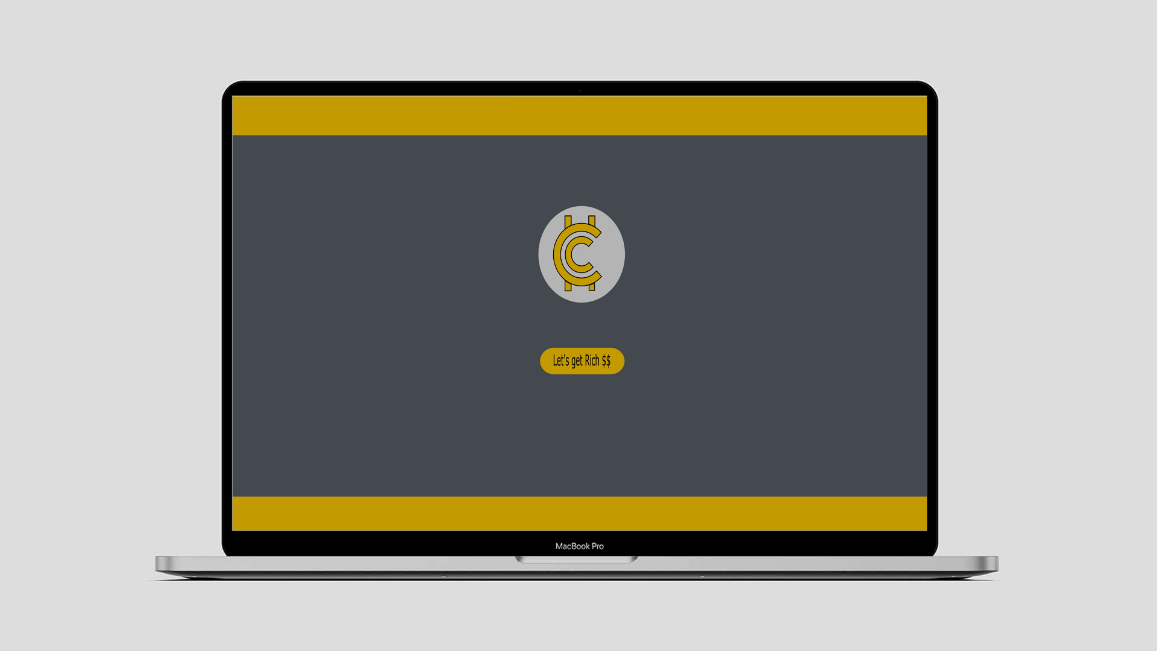


Figura 2-Landing Page

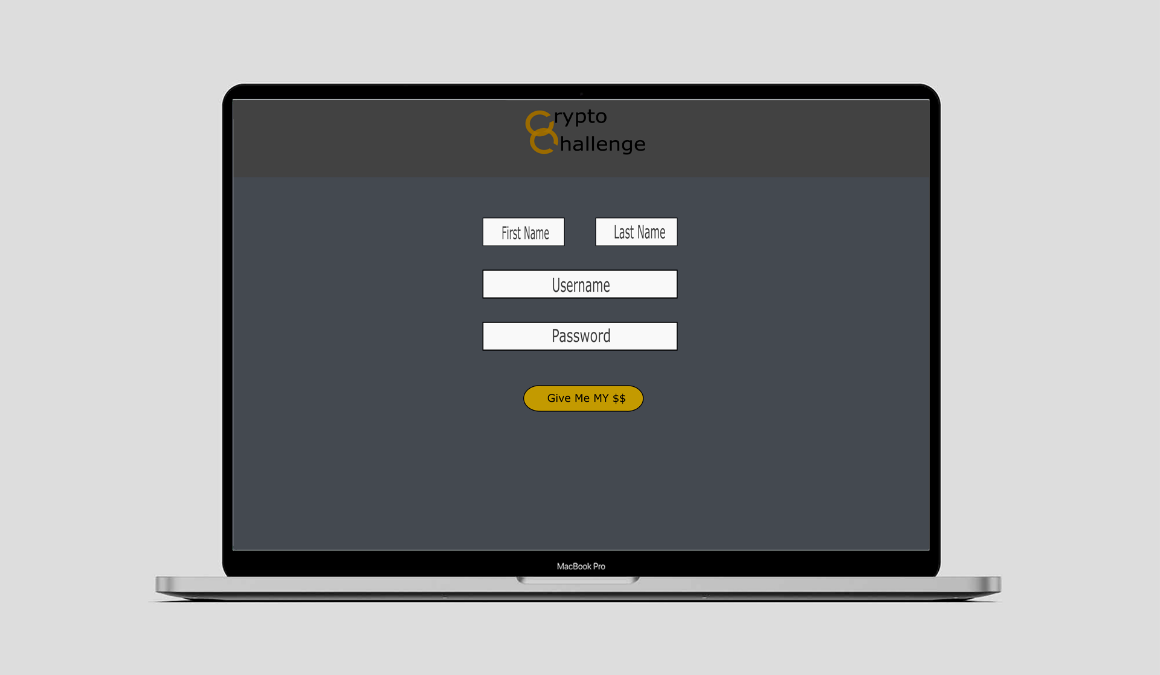


Figura 3- LogIn Page



Figura 4- Wallet Page

Figura 5- Home Page