Τεχνολογικό Υπόβαθρο

- Blockchain:
- Ethereum:
- Smart Contract:
- Server:
- JWT:
- SQLite:
- SHA224:
- API:
- Hash:
- Query:
- HS256:
- Hexadecimal Seed:
- JSON-RPC:
- React:

Συντομογραφίες

- F/E - f/e: Front end, το κομμάτι της εφαρμογής με το οποίο διαδρά ο χρήστης
- B/E - b/e: Back end, το κομμάτι της εφαρμογής που υλοποιεί βασικές λειτουργίες οι οποίες σχετίζονται με την βάση δεδομένων
- CRUD: Create / Read / Update / Delete
- JWT: JSON Web Tokens
- ETH: Ethereum
- BTC: Bitcoin
- XRP: Ripple
- ORM: Object Relational Mapper

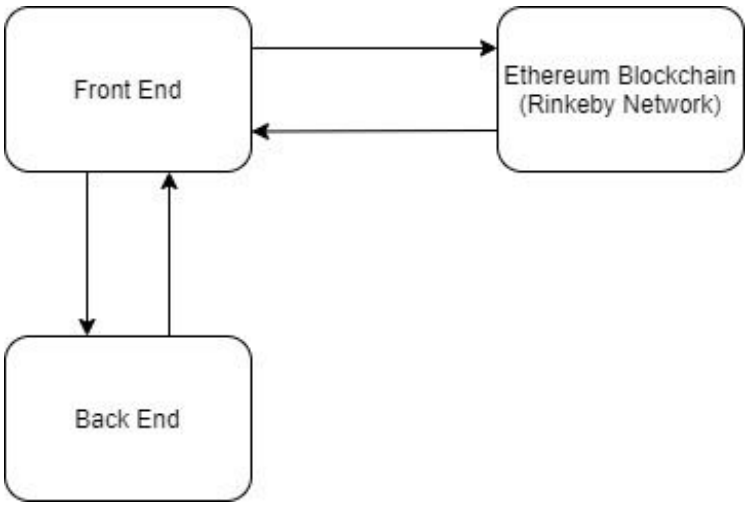
- API: Application Programming Interface

Εφαρμογή GreenWallet

Το GreenWallet είναι μια εφαρμογή η οποία αποτελεί ένα ψηφιακό πορτοφόλι το οποίο δίνει την δυνατότητα στους χρήστες του να εκτελούν συναλλαγές με καταστήματα τα οποία επιτρέπουν την πληρωμή μέσω του Ethereum Blockchain. Οι συναλλαγές πραγματοποιούνται με την μεταφορά χρημάτων από το πορτοφόλι του χρήστη στο smart contact του αντίστοιχου μαγαζιού. Επιπλέον υποστηρίζονται συναλλαγές μεταξύ χρηστών, από πορτοφόλι σε πορτοφόλι χωρίς την χρήση smart contact.

// Η φωτογραφία θα αλλάξει με την τελευταία έκδοση

Αρχιτεκτονική Υψηλού Επιπέδου



Για την λειτουργία της εφαρμογής, το κομμάτι με το οποίο διαδρά ο χρήστης πρέπει να επικοινωνεί με το b/e και το Ethereum blockchain. Το b/e αναλαμβάνει κυρίως το βάρος της ταυτοποίησης του χρήστη και της αποθήκευσης βασικών δεδομένων για την ομαλή λειτουργεία της εφαρμογής. Με την επικοινωνία με το Ethereum blockchain γίνεται δυνατή η μεταφορά χρημάτων σε κάποιο smart contract ή σε άλλο πορτοφόλι χρήστη του GreenWallet. Η μεταφορά χρημάτων σε δημόσια διεύθυνση που δεν ανήκει στο GreenWallet είναι επίσης δυνατή. Πιο συγκεκριμένα:

Λειτουργίες που εκτελεί το b/e είναι:

- Αποθήκευση βασικών πληροφοριών για τα καταστήματα / smart contracts που έχει αποθηκεύσει ο χρήστης
- CRUD στα δεδομένα των καταστήματα / smart contracts
- Αποθήκευση αναγκαίων πληροφοριών για την ταυτοποίηση του χρήστη
- Ταυτοποίηση του χρήστη
- Δημιουργία “session” για την ομαλότερη λειτουργία της εφαρμογής και την καλύτερη εμπειρία περιήγησης του χρήστη (JWT)
- Ενημέρωση της εφαρμογής με τις πιο πρόσφατες συναλλαγματικές αξίες των τριών πιο χρησιμοποιημένων κρυπτονομισμάτων ETH, του BTC και του XRP

- Παραγωγή mnemonic για την δημιουργία καινούργιων πορτοφολιών

Λειτουργίες που εκτελούνται στο Blockchain:

- Μεταφορά χρημάτων από πορτοφόλι σε πορτοφόλι μεταξύ χρηστών χωρίς την μεσολάβηση smart contract
- Πληρωμές σε καταστήματα που υποστηρίζουν την πληρωμή μέσω του GreenWallet
- Ενημέρωση για το υπόλοιπο του λογαριασμού του χρήστη

Λειτουργίες του f/e:

- Σύνδεση και προβολή όλων των λειτουργιών που προσφέρουν το b/e και το blockchain
- Δημιουργία μιας ευχάριστης εμπειρίας για τον χρήστη

Ανάλυση υλοποίησης των λειτουργιών της εφαρμογής GreenWallet

Αποθήκευση αναγκαίων πληροφοριών για την ταυτοποίηση του χρήστη:

Η διαδικασία της αποδοχής των στοιχείων, τα οποία εισάγει ο χρήστης, και η εισαγωγή τους στην βάση γίνεται στο b/e κομμάτι της εφαρμογής. Η εφαρμογή χρησιμοποιεί την SQLite για την διαχείριση της σχεσιακής βάσης δεδομένων στην οποία αποθηκεύονται όλα τα δεδομένα της εφαρμογής. Για την ευκολότερη ενσωμάτωση και χρήση της βάσης δεδομένων, χρησιμοποιείται ένα ORM το SQLAlchemy (https HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/"// HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/"www HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/". HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/"sqlalchemy HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/". HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/"org HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/" HYPERLINK "https://www.sqlalchemy.org/"/).

Στην βάση δεδομένων ένας χρήστης αναπαρίσταται από τα παρακάτω στοιχεία:

- **Id**: Ένας μοναδικός ακέραιος αριθμός ο οποίος ξεχωρίζει τον χρήστη από όλους τους άλλους χρήστες. Ο αριθμός αυτός συμπληρώνεται απευθείας από την βάση δεδομένων κατά την δημιουργία του χρήστη. Το Id αποτελεί το κύριο κλειδί του πίνακα.
- **Username**: Το όνομα με το οποίο αναπαρίσταται ο χρήστης μέσα στην εφαρμογή. Δύο χρήστες δεν επιτρέπεται να έχουν το ίδιο Username.
- **Password**: Το συνθηματικό του χρήστη, το οποίο είναι αναγκαίο για την ταυτοποίηση του στην εφαρμογή. Ο κάθε χρήστης μπορεί να επιλέξει το συνθηματικό του κατά την δημιουργία του λογαριασμού του. Για την μεγαλύτερη ασφάλεια των δεδομένων των χρηστών, κανένα συνθηματικό δεν αποθηκεύεται στην βάση δεδομένων. Στην βάση δεδομένων αποθηκεύεται το αποτέλεσμα που προκύπτει από την εφαρμογή της συνάρτησης κατακερματισμού SHA224 στον κωδικό που παρέχει ο χρήστης.
- **Mnemonic**: Μια σειρά 12 τυχαίων λέξεων, στην αγγλική γλώσσα, οι οποίες χρησιμοποιούνται για την παραγωγή του ιδιωτικού κλειδιού του πορτοφολιού του χρήστη. Παραδείγματος χάριν: ‘almost spice garment loop slight blanket copper sun empty cement work sail’. Το Mnemonic παράγεται αυτόματα από την εφαρμογή, όταν ο χρήστης δημιουργεί τον λογαριασμό του.

```
class User(base):
    __tablename__ = 'Users'

    Id = Column(Integer, primary_key=True, autoincrement=True)
    Username = Column(String)
    Password = Column(String)
    Mnemonic = Column(String)
```

// Κλάση η οποία περιγράφει την δομή του πίνακα του χρήστη στο ORM

Οι βασικές λειτουργίες που υλοποιούνται πάνω στα δεδομένα του χρήστη είναι:

- **Η δημιουργία νέου χρήστη:**

  Η δημιουργία ενός νέου χρήστη γίνεται με την υποβολή της αντίστοιχης φόρμας στο f/e κομμάτι της εφαρμογής. Κατά την υποβολή γίνεται χρήση του εκτεθειμένου API υπεύθυνου για την δημιουργία νέων χρηστών, το οποίο βρίσκεται (τοπικά) στο 'http://127.0.0.1:5000/createUser/'. Ο κώδικας που τρέχει με την κλήση του API είναι υπεύθυνος για την υποδοχή των δεδομένων, την παραγωγή ενός μνημονικού, την δημιουργία του hash που θα αποθηκευτεί στην θέση του συνθηματικού του χρήστη και τελικά την αποθήκευση των δεδομένων στην βάση. Για την συγγραφή πιο κατανοητού αλλά και επεκτάσιμου κώδικα, όλες αυτές οι λειτουργίες δεν εκτελούνται

μέσα σε μια μέθοδο αλλά σε πολλές μεθόδους οι οποίες εκτελούν μια συγκεκριμένη λειτουργία.

<u>Ταυτοποίηση του χρήστη</u>:

Η διαδικασία της ταυτοποίησης του χρήστη ξεκινάει με το που μπει στην εφαρμογή. Η διαδικασία αυτή διαχωρίζεται σε δύο κυρίως βήματα:

- Τον έλεγχο των δεδομένων που εισάγει ο χρήστης
- Την έκδοση ενός JSON Web Token το οποίο επιστρέφεται στον χρήστη

- Για να μπορέσει ένας χρήστης να χρησιμοποιήσει την εφαρμογή, πρέπει πρώτα είτε να συνδεθεί στον ήδη υπάρχον λογαριασμό του ή να φτιάξει έναν καινούργιο και να συνδεθεί σε αυτόν. Ο χρήστης παρέχει τα στοιχεία του σε μια φόρμα στο f/e κομμάτι της εφαρμογής τα οποία με την κλήση του αντίστοιχου API, τοπικά στο `'http://127.0.0.1:5000/login/',` μεταφέρονται στο b/e. Ο κώδικας που τρέχει με την κλήση του API είναι υπεύθυνος για:
  - Να μετατρέψει το συνθηματικό που έχει εισάγει ο χρήστης στην φόρμα και έχει περαστεί στις παραμέτρους της κλήσης του API σε hash. Για τον κατακερματισμό του συνθηματικού (SHA224) είναι υπεύθυνη η μέθοδος `passwordHashing(password)` η οποία κάνει χρήση της hashlib βιβλιοθήκης της python και πιο συγκεκριμένα της μεθόδου `sha224(password)` (python docs : https://docs.python.org/3/library/hashlib.html)
  - Μετά την επιστροφή του hash του συνθηματικού, γίνεται ένα query στην βάση, το οποίο μας επιστρέφει όλες τις εγγραφές που υπάρχουν, φιλτραρισμένες με το Username και το Hash. Είναι πιθανό να γυρίσει μια ή καμία εγγραφή μιας και το Username είναι μοναδικό. Η περίπτωση που δεν υπάρχει καμία εγγραφή μας υποδεικνύει ότι ο χρήστης έχει εισάγει λάθος στοιχεία. Στην περίπτωση που επιστραφεί μια εγγραφή, σημαίνει ότι ο χρήστης έχει πληκτρολογήσει σωστά τα στοιχεία του και επομένως το σύστημα συνεχίζει με την έκδοση ενός JSON Web Token
- Για την έκδοση του JSON Web Token γίνεται χρήση της βιβλιοθήκης `pyjwt` (https://pyjwt.readthedocs.io/en/latest/). Όλες οι λειτουργίες που συσχετίζονται με το JSON Web Token βρίσκονται στο αρχείο JWTTokenUtils.py. Οι βασικοί μέθοδοι που περιέχονται είναι οι εξής:
  - `createToken(userId)`: Η μέθοδος δέχεται ως όρισμα το Id του χρήστη που επιστράφηκε κατά τον έλεγχο των δεδομένων του, το προσθέτει στο ωφέλιμο φορτίο και με την χρήση ενός μυστικού κλειδιού δημιουργεί το JSON Web Token. Κάθε token δίνει την δυνατότητα στον χρήστη να αποδεικνύει την ταυτότητα του για 20 λεπτά μετά την έκδοση του ή μέχρι αυτός να κλείσει τον περιηγητή του. Για την δημιουργία του token χρησιμοποιείται ο συμμετρικός αλγόριθμος HS256 (HMAC with SHA-256).

- checkToken(token): Η μέθοδος δέχεται ως όρισμα το JSON Web Token που παρέχει ο χρήστης με κάθε API call μετά την είσοδο του στο σύστημα και ελέγχει αν αυτό είναι έγκυρο. Δυο βασικοί λόγοι που μπορεί ένα token να μην είναι έγκυρο είναι να έχει λήξει ή να έχει τροποποιηθεί από κάποιον κακόβουλο χρήστη. Και στις δυο περιπτώσεις το token απορρίπτεται από το σύστημα.

```
def createToken(userId):
    # Create JWT token
    payload = {
        'exp': datetime.datetime.utcnow()+datetime.timedelta(minutes=20),
        'iat': datetime.datetime.utcnow(),
        'sub': userId
    }

    token = jwt.encode(payload, SECRET_KEY, algorithm='HS256')

    return token
```

// The method responsible for creating the JSON Web Token

```
def checkToken(token):
    # Decoding the token
    try:
        payload = jwt.decode(token, SECRET_KEY)
        return payload['sub']
    except jwt.ExpiredSignatureError:
        print('Token Expired')
        return None
    except jwt.InvalidTokenError:
        print('Invalid Token')
        return None
```

// The method responsible for checking the JSON Web Token

<u>Ενημέρωση της εφαρμογής με τις πιο πρόσφατες συναλλαγματικές αξίες των τριών πιο χρησιμοποιημένων κρυπτονομισμάτων</u>:

Η συναλλαγματική αξία του ETH δεν είναι σταθερή, γεγονός που σημαίνει ότι η αξία των χρημάτων του χρήστη αλλάζει συνεχώς. Ο χρήστης πρέπει να είναι ικανός να γνωρίζει την συναλλαγματική αξία του ETH ώστε να μπορεί να υπολογίσει πότε είναι προτιμότερο να προσθέσει ή να αφαιρέσει χρήματα από το λογαριασμό του. Οι τιμές του ETH, του BTC και του XRP γνωστοποιούνται στο σύστημα με ένα API call στο CoinLayer API (https://coinlayer.com/). Το API call θα ήταν δυνατό να γίνει απευθείας στο f/e κομμάτι της εφαρμογής, παρ' όλ' αυτά επειδή χρησιμοποιείται ένα μυστικό κλειδί το οποίο πρέπει να παραμείνει κρυφό, γίνεται στο b/e. Οι πιο πρόσφατες τιμές επιστρέφονται στο b/e και από εκεί αποστέλλονται στο f/e όπου και εμφανίζονται στην αρχική σελίδα του χρήστη.

```
def getExchangeRates():
        #
        # Making an API call to the CoinLayer API
        # to get the current exchange rates
        #
        data = request.args.to_dict()

        coin = data['target']
        COINLAYER_URL = 'http://api.coinlayer.com/live'
        COINLAYER_API_KEY = 'eb3c76bbef63fe4aa71b802bc25bef0c'
        PARAMS = { 'access_key': COINLAYER_API_KEY, 'target': coin }
        HEADERS = {'content-type':'application/json'}

        response = requests.get(url=COINLAYER_URL, params=PARAMS,
                                headers=HEADERS)
        data = json.loads(response.text)

        dataToSent = {'ETH': data['rates']['ETH'],
                 'BTC': data['rates']['BTC'],
                 'XRP': data['rates']['XRP']
        }

        responseToReact = flask.Response(json.dumps(dataToSent))
        responseToReact.headers['Access-Control-Allow-Origin'] = '*'
        return responseToReact
```

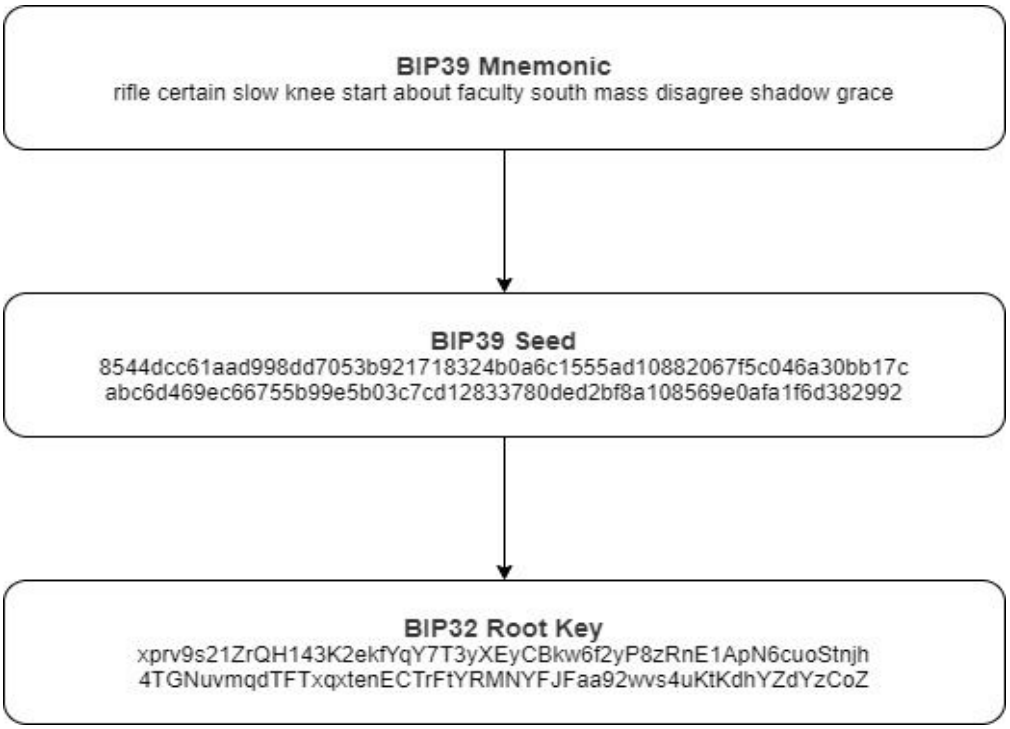
Παραγωγή mnemonic για την δημιουργία καινούργιων πορτοφολιών

Όπως προαναφέρθηκε στο κεφάλαιο 'Αποθήκευση αναγκαίων πληροφοριών για την ταυτοποίηση του χρήστη' μια από τις βασικές ευθύνες της εφαρμογής κατά την δημιουργία ενός νέου χρήστη, είναι η παραγωγή ενός μνημονικού το οποίο αποθηκεύεται στην βάση δεδομένων μαζί με όλα τα υπόλοιπα στοιχεία του χρήστη. Η χρήση του μνημονικού αρχικά προτάθηκε στο Bitcoin Improvement Proposal 39 (https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki) και έκτοτε υλοποιείται στις περισσότερες εφαρμογές Blockchain πορτοφολιών. Το μνημονικό είναι πολύ χρήσιμο μιας και από αυτό μπορεί να παραχθεί ένα 512-bit hexadecimal seed, από το οποίο μετά μπορούν να παραχθούν τα ιδιωτικά και τα δημόσια κλειδιά του λογαριασμού. Ένα μεγάλο όφελος της χρήσης του μνημονικού έναντι της χρήσης μόνο του hexadecimal seed, είναι ότι είναι πολύ πιο εύκολο να απομνημονευθεί από τον χρήστη μιας και είναι ουσιαστικά μια σειρά από 12-24 λέξεις. Στην εφαρμογή GreenWallet γίνεται η χρήση μνημονικών των 12 λέξεων. Για την παραγωγή του μνημονικού χρησιμοποιείται η μέθοδος `generate_mnemonic()` της κλάσης `wallet` από την βιβλιοθήκη της python pywallet (https://pypi.org/project/pywallet/).

```
from pywallet import wallet

class walletUtils:

    @staticmethod
    def createSeed():
        return wallet.generate_mnemonic()
```



Αρχικοποίηση του πορτοφολιού

Αμέσως μετά την επιτυχή σύνδεση του χρήστη στην εφαρμογή, καλείται η μέθοδος initialiseWallet(). Σε αυτή την μέθοδο γίνεται η δημιουργία του βασικού αντικειμένου της βιβλιοθήκης της JavaScript Web3. Όλες οι λειτουργίες που σχετίζονται με την διάδραση της εφαρμογής με το Ethereum Blockchain βασίζονται στην χρήση αυτού του αντικειμένου (https://web3js.readthedocs.io/en/v1.2.11/web3.html). Για την δημιουργία του αντικειμένου χρειάζεται να περάσουμε ως όρισμα έναν προμηθευτή (Provider). Ο provider είναι το μέσο με το οποίο η εφαρμογή «μιλάει» με το Ethereum Blockchain και είναι υπεύθυνο για την υποδοχή JSON-RPC αιτημάτων και την επιστροφή της απάντησης τους. Το GreenWallet χρησιμοποιεί έναν HTTP provider, ο οποίος δημιουργείται με την χρήσης της βιβλιοθήκης της JavaScript @truffle/hdwallet-provider (https://www.npmjs.com/package/@truffle/hdwallet-provider). Επίσης γίνεται η χρήση ενός Infura Ethereum node έναντι της χρήσης ενός προσωπικού Ethereum node. Κάποια από τα πλεονεκτήματα της χρήσης του Infura Ethereum node είναι:

- Ταχύτητα
- Επεκτασιμότητα

Μετά την δημιουργία του web3 αντικειμένου, η εφαρμογή ανακτά την διεύθυνση του λογαριασμού του χρήστη, την οποία και αποθηκεύει για μετέπειτα χρήση στο state του React Web Application.

Ενημέρωση για το υπόλοιπο του λογαριασμού του χρήστη

Μια πολύ σημαντική δυνατότητα, η οποία είναι απαραίτητη για κάθε εφαρμογή wallet, είναι η ενημέρωση του χρήστη για το υπόλοιπο του αμέσως όταν αυτό αλλάζει. Το υπόλοιπο του ο χρήστης μπορεί να το δει στην αρχική σελίδα της εφαρμογής. Για την ανάκτηση του υπολοίπου του χρήστη γίνεται η χρήση της μεθόδου (web3.eth.getBalance(accountAddress)), του web3 αντικειμένου (Περισσότερες πληροφορίες στο μέρος «Αρχικοποίηση του πορτοφολιού») και καλείται πάντα αμέσως μετά την κλήση μεθόδων οι οποίες μπορεί να έχουν ως αποτέλεσμα την αλλαγή της κατάστασης του Blockchain και συγκεκριμένα της αλλαγής του υπολοίπου του λογαριασμού του χρήστη. Λειτουργιές που αλλάζουν το υπόλοιπο του χρήστη είναι η μεταφορά χρημάτων από την χρεωστική του κάρτα στο πορτοφόλι του, η μεταφορά χρημάτων σε πορτοφόλια άλλων χρηστών και τέλος η μεταφορά χρημάτων σε καταστήματα μέσω των smart contracts τους.