**<Leprecoin>**

מסמך עיצוב

<אילן נוה>

<גרסה 1>

<תאריך>

**היסטוריית גרסאות המסמך**

|  |  |  |
| --- | --- | --- |
| **תאריך** | גרסה | **תקציר השינויים** |
| 28.5.19 | 1 |  |
|  |  |  |
|  |  |  |
|  |  |  |



**1. הקדמה**

החלק הזה ישמש להצגה כללית של מסמך העיצוב. הסבר כללי, וקישור למסמכים קודמים כמו מסמך האפיון עליו אנו מתבססים

1.1 מטרה

*המטרה של המסמך היא להסביר על הפרויקט, על עיצובו ועל איך הפרוטוקול בוא עובד.*

1.2 המוצר

* *המוצר נקרא Leprecoin והוא מטבע דיגיטלי מאובזר קריפטוגרפי*
* *המוצר מאפשר העברת כספים בטוחה וחוקית מסוג המטבע על ידי ממשק ארנק*

1.3 קישור למסמכים קודמים

*ראה מסמך אפיון*

*ראה מסמך חקר*

1.4 הגדרות

*מטבע דיגיטלי ( (Cryptocurrency- המטבע הדיגיטלי מונפק על ידי גוף פרטי כלשהו שמפתח טכנולוגיה שאינה מאפשרת לשכפל אותו. המטבע בדרך כלל אינו שייך לגוף שהנפיק אותו אלא פתוח לכולם. למעשה מדובר בפרוטוקול (תוכנה), שרץ במקביל על אלפי מחשבים ונרשם על ידיהם. מטבע דיגיטלי קריפטוגרפי הוא מטבע שאבטחתו נעשת באמצעות מפתחות פרטיים וציבוריים.*

*בלוק צ'יין - היא תפיסה טכנולוגית במחשוב המאפשרת פעילות עסקית מאובטחת באינטרנט ואימות של טרנזקציות עסקיות בין צדדים שונים ללא צורך בישות ניהול מרכזית. הבלוקצ'יין היא בעצם שרשרת של בלוקים כשכל בלוק מקושר לקודמו ולכל בלוק יש קוד hash יחודי.*

*ביטקוין - הביטקוין הוא מטבע שאינו תלוי בגוף מרכזי. במקום זאת, ביטקוין הוא רשומה בקובץ ציבורי המכונה "שרשרת בלוקים" (בלוקצ'יין), ניתן לנהל אותו או להשתמש בו בעזרת תוכנה המכונה "ארנק". מערכת הביטקוין היא מערכת המבטיחה שהתשלומים יתבצעו באופן תקין כך שסכום שהועבר בתשלום אכן נגרע מארנק המקור והתווסף לארנק היעד.*

*כרייה – הוספת בלוק לשרשרת הבלוקים על ידי תהליך מסובך הכולל חישובים מתמטיים ודורש יכולות מחשוב גדולות*

*מפתח פרטי- מפתח פרטי השייך למי שיצר אותו ורק הוא יכול להשתמש בוא על מנת לחתום מידע.*

*מפתח ציבורי- לכל מפתח פרטי יש מפתח ציבורי שמטרתו שיהיה ידוע לכולם, בעזרת המפתח הציבורי אפשר לאמת את החותם על חתימות דיגיטלות. אי אפשר לחשב את המפתח הפרטי מהציבורי*

*חתימה דיגיטלית- מידע אשר עבר עלי שימוש במפתח פרטי (נחתם על ידו) נקרא חתימה. בעזרת מפתח הציבורי אפשר לאמת אם החותם הוא בעל המפתח הפרטי שמתאים לציבורי.*

*P2P- בניגוד לתקשורת הנפוצה כיום- בין שרת ללקוח. p2p זאת שיטת תקשורת שלא תלויה בשרת שמנהל אותה אלא כל המשתמשים במערכת מקימים אותה. מטבע המשתמש ברשת כזאת נקרא מטבע מאובזר*

1. ארכיטקטורת המערכת

חלק זה כולל את תיאור מבנה המערכת ופירוט המודולים השונים בה

2.1 מבט על

המערכת מחולקת לשלושה חלקים עיקריים: למבנה הנתונים של הבלוקצ'יין, לארנק המשתמש הכולל ממשק גרפי ולמימוש התקשורת באמצעות node.

הבלוקצ'יין היינו שרשרת הבלוקים שמכילה בתוכה את כל העסקאות שנעשו במטבע. בעזרת הפרוטוקול תקשורת לכל המשתמשים ברשת יש אותו עותק של בלוקצ'יין ובכך הם יכולים לאמת עסקאות ולאשר עסקאות חוקיות.

הבלוקצ'יין עצמו הוא רשימה של עצמים מסוג בלוק כאשר כל בלוק מכיל את הhash של קודמו, מידע נוסף (יורחב בהמשך) ואת רשימת העסקאות של אותו הבלוק. כל הבלוקצ'יין מכיל כך, את כל העסקאות שנעשו עם המטבע. הבלוקצ'יין עטוף במחלקה שמקשרת אותו למבנה הנתונים, בוא הוא נשמר בטבלאות בשפת sql. השרשרת נשמרת בטבלאות על מנת לאפשר גישה נוכה לכל בלוק לסידור מתאים של הנתונים. כל ארנק שמכיל את הבלוקצ'יין הוא בעצם מכיל את המחלקה שעוטפת אותו עם בסיס הנתונים ובכך יש לו גישה למידע דרך בסיס הנתונים.

בתחילת ההרצה התוכנה יוצרת את עצם מסוג זה פעם אחת ומעבירה אותו לארנק ולnode.

הארנק הינו הדרך להשתמש במטבע. הארנק מאפשר להעביר את המטבע לארנקים אחרים, ולראות את המטבעות שאתה מקבל. הארנק כולל ממשק גרפי על מנת לאפשר למשתמש להשתמש בוא. הארנק מכיל בתוכו את הבלוקצ'יין, עם גישה לבסיס הנתונים, הוא מוצא על פי המפתח הפרטי שבקובץ את הכתובת של במשתמש, על הפי הבלוקצ'יין והכתובת הוא יכלו למצוא את כמות הכסף שיש לו (ראה מבנה עסקה ומסמך חקר). בנוסף לכך יש לו גישה לעצם מסוג node שאחראי על התקשורת ועל אימות עסקאות.

תקשורת מסוג p2p נבנית על כך שכל מחשב (peer או node) משמשים גם כשרת וגם כלקוח. כל node בתקשורת מסוג זה חייב להתנהג על פי אותם חוקים על מנת שהרשת תהיה אמינה, לכן הם נקראים גם peers (=עמיתים).

לכן על מנת לממש זאת התקשורת בנויה על שלוש מחלקות. המחלקה הראשונה היא השרת, והוא מאפשר על ידי ריצה בthread שונה לקבל כל הזמן הודעות ממשתמשים אחרים. תפקידו העיקרי של השרת הוא לאחסן את ההודעות כך שיהיה גישה אליהם. המחלקה השנייה היא הלקוח, הלקוח הוא בעצם שולח את ההודעות לכתובת שמצוינת לו או לכל הnodes שהוא מודע לקיומם. המחלקה השלישית והמרכזית היא המחלקה node שהיא מריצה את השרת, ואת הלקוח כשצריך. לכן, היא אחראית למימוש של כל peer ולתפקוד התקין שלו. בנוסף לכך, המחלקה מחזיקה את העצם של הבלוקצ'יין (העצם עם מבנה הנתונים) ובכך היא יכולה לאמת עסקאות ובלוקים שהיא מקבלת.

2.2 פירוט רכיבי המערכת

\*כשכתוב input ו outputבתיאור המחלקות הבאות, הכוונה לא לקלט מהמשתמש ולפלט של תוכנה אלא לinput ולoutput של עסקה. (ראה במחלקות ובמסמך חקר)

\*את התפקידים של הinputs והoutputs ניתן לראות במסמך חקר.

\*הכוונה בפונקציית hash היא בדרך כלל SHA256 למעט, השימוש בRIPEMD160 ביצירת הכתובת של הארנק. (ראה בספרייה pycrypto).

|  |
| --- |
| Block |
| תכונות   * self.number - המספר הסודר של הבלוק * self.nonce – המספר שמחושב עם הערך של הhash על מנת ליצור proof of work * self.prev – ערך הhash של הבלוק הקודם * self.difficulty – ערך הקושי ביצירת proof of work * self.transactions – העסקאות שבבלוק * self.time\_stamp – חותמת הזמן של יצירת הבלוק * self.hash\_code – קוד הhash של הבלוק |
| פעולות   * \_\_init\_\_ - יוצרת את הבלוק, מקבלת את הערכים של כל התכונות של הבלוק. או משתמשת בערכים קבועים.   השימוש בערכים אלו הוא בשביל ליצור בלוק חדש ולא כשמוציאים אותו ממבנה הנתונים   * \_\_str\_\_ - ממירה את הבלוק למחרוזת ומחזירה אותו * hash\_block – מחשבת את ערך הhash של הבלוק * hash\_transactions – מחשבת ומחזירה את ערך הhash של כל העסקאות ביחד * add\_transaction – מקבלת עסקה ומוסיפה אותה לרשימת העסקאות בבלוק * mine\_block – הפעולה כורת את הבלוק על ידי חישוב ערך nonce מתאים * is\_valid\_proof – מחזירה אם יש proof of work תקין על הבלוק * serialize – מחזירה tuple של הבלוק על מנת שיוכל להיכנס בקלות לdatabase * deserialize – פעולה סטאטית שמחזירה עצם בלוק מערך של serialized בלוק ורשימת העסקאות שבו |

|  |
| --- |
| Transaction |
| תכונות   * self.inputs – רשימת הinputs בעסקה * self.outputs – רשימת הoutputs של העסקה * self.transaction\_id – ערך הhash של העסקה |
| פעולות   * \_\_init\_\_ - יוצרת את העסקה. מקבלת רשימה של inputs וoutputs * add\_input – מקבלת input מוסיפה אותו לרשימת הinputs * \_\_str\_\_ - ממירה את העסקה למחרוזת ומחזירה אותה * add\_output – מקבלת output ומוסיפה אותו לרשימת הoutputs * hash\_transaction – מחשבת ומחזירה את ערך הhash של העסקה * is\_valid\_proof – מחזירה אם יש proof of work תקין על הבלוק * deserialize – פעולה סטאטית שמחזירה עצם עסקה מרשימת inputs וoutputs |

|  |
| --- |
| Output |
| תכונות   * self.value – מספר הכסף שמעבירים בעסקה * self.address – כתובת הארנק שאליה מעבירים (לא ip) |
| פעולות   * \_\_init\_\_ - יוצרת את הoutput. מקבלת את התכונות של המחלקה * \_\_str\_\_ - ממירה את הoutput למחרוזת ומחזירה אותה * hash\_output – מחשבת ומחזירה את ערך הhash של הoutput * serialize – מחזירה tuple של הoutput על מנת שיוכל להיכנס בקלות לdatabase * deserialize – פעולה סטאטית שמחזירה עצם output מ- serialized output |

|  |
| --- |
| Input |
| תכונות   * self.transaction\_id – ערך הhash של העסקה שהinput משתמש בה להוכחה * self.output\_index – המספר הסידורי של הoutput להוכחה בעסקה שממנה מוכיחים * self.proof – ההוכחה ליכולת להשתמש בoutput. ההוכחה היא רשימה המכילה בתוכה את החתימה והמפתח הציבורי של החותם. |
| פעולות   * \_\_init\_\_ - יוצרת את הinput. מקבלת את כל התכונות של המחלקה * \_\_str\_\_ - ממירה את הinput למחרוזת ומחזירה אותה * hash\_input – מחשבת את ערך הhash של הinput ומחזירה אותו * add\_transaction – מוסיפה עסקה לרשימת העסקאות בבלוק * mine\_block – הפעולה כורת את הבלוק על ידי חישוב ערך nonce מתאים * is\_valid\_proof – מחזירה אם יש proof of work תקין על הבלוק * serialize – מחזירה tuple של הinputעל מנת שיוכל להיכנס בקלות לdatabase * deserialize – פעולה סטאטית שמחזירה עצם input מערך שלinput serialized |

|  |
| --- |
| UnspentOutput |
| תכונות   * self.output – הoutput הלא מבוזבז * self.transaction\_id – המזהה של העסקה ממנה לקוח הoutput * self.output\_index – המספר הסידורי של הoutput בעסקה ממנה הוא לקוח |

|  |
| --- |
| BlockChain |
| תכונות   * self.chain – רשימת הבלוקים * self.transaction\_pool – רשימת העסקאות שצפויות להיכנס לבלוק החד שייווצר בתהליך הכרייה * self.logger – עצם מסוג logger (ראה ספרייה logging) בשביל הפעולות במחלקה |
| פעולות   * \_\_init\_\_ - יוצרת את הבלוקצ'יין. מקבלת logger ורשימת בלוקים ויוצרת את העצם עם רשימת עסקאות ריקה * add\_new\_block – הפעולה מקבלת את כתובת הארנק של הכורה, כורת בלוק חדש ומוסיפה אותו לשרשרת עם תשלום לכורה * \_\_str\_\_ - ממירה את הבלוק למחרוזת * add\_transaction – מוסיפה עסקה לרשימת העסקאות בשרשרת |

|  |
| --- |
| BlockChainDB |
| המחלקה יורשת מהמחלקה BlockChain  תכונות   * self.logger – עצם מסוג logger בשביל הפעולות במחלקה * self.connection – העצם שאחראי לחיבור של המחלקה למבנה הנתונים (ראה ספרייה sqlite3) * self.cursor – המצביע שנוצר בחיבור, אחראי על ביצוע הפעולות השונות במבנה הנתונים * self.path – מיקום קובץ מבנה הנתונים * עצמים של הטבלאות השונות במבנה הנתונים: טבלת בלוקים, טבלת עסקאות, טבלת inputs וטבלת outputs |
| פעולות   * \_\_init\_\_ - יוצרת את העצם המכיל בתוכו גם את העצם המורש. מקבלת כפרמטרים את עצם הlogger ואת השם של מבנה הנתונים (אם לא מועבר שם אז יש לו ערך קבוע). * create\_connection – יוצר חיבור חדש למבנה הנתונים * close\_connection – סוגר את החיבור למבנה הנתונים * insert\_block – מקבלת בלוק ומכניסה אותו למבנה הנתונים * extract\_block – מקבלת מספר שורה, ומחזירה את הבלוק ממספר שורה זה * extract\_inputs – הפעולה מקבלת את הid של עסקה ומחזירה את כל הinputs השייכים לעסקה זאת * extract\_outputs – הפעולה מקבלת את הid של עסקה ומחזירה את כל הoutputs השייכים לעסקה זאת * extract\_transactions – הפעולה מקבלת מספר בלוק ומחזירה את כל העסקאות השייכות לבלוק זה * extract\_chain – הפונקציה מוציאה ממבנה הנתונים את כל שרשרת הבלוקים * add\_new\_block\_to\_db – הפונקציה מוסיפה לבנה נתונים בלוק חדש שעובר כרייה. הפונקציה מקבלת את הכתובת של הכורה * update\_chain – הפונקציה מעדכנת את רשימת הבלוקים בהתאם למבנה הנתונים * add\_downloaded\_blocks – הפונקציה מקבלת בלוקים שהורדו מהאינטרנט ומוסיפה אותם למבנה הנתונים ולרשימה של הבלוקים |

|  |
| --- |
| Table |
| תכונות   * self.connection – החיבור למבנה הנתונים שאליו שייכת הטבלה * self.curser – המצביע שנוצר בחיבור, אחראי על ביצוע הפעולות השונות במבנה הנתונים * self.table\_name – שם הטבלה * self.structure- מבנה הטבלה |
| פעולות   * \_\_init\_\_ - יוצרת את עצם הטבלה. מקבלת את כל התכונות של המחלקה * create\_table – יוצר את הטבלה אם היא לא קיימת במבנה הנתונים * insert – מקבלת tuple של עצם ומכניסה אותו לטבלה |

|  |
| --- |
| CryptoSet |
| תכונות   * self.private\_key – המפתח הפרטי בסט * self.public\_key – המפתח הציבורי בסט |
| פעולות   * \_\_init\_\_ - יוצרת את הסט. מקבלת את הערך של המפתח הפרטי * sign – מקבלת קוד hash חותמת עליו ומחזירה את החתימה * decrypt– הפעולה מקבלת מידע מוצפן ומפענחת אותו באמצעות המפתח הפרטי * encrypt – פעולה סטאטית המקבלת מפתח ציבורי והודעה להצפין, מצפינה אותה ומחזירה את ההודעה המוצפנת. * verify – פעולה סטאטית המקבלת מפתח ציבורי, קוד hash וחתימה, ומאמתת את החתימה. מחזירה אמת אם היא אמיתי ושקר אחרת. * hash – הפעולה מקבלת הודעה מעבירה עליה פונקצית hash ומחזירה את התוצאה |

|  |
| --- |
| Wallet |
| המחלקה יורשת מהמחלקה CryptoSet  תכונות   * self.address – כתובת הארנק. מתקבלת על ידי העברת פונקציית sha256 על המפתח הציבורי שלו, ועל התוצאה העברת פונקציה ripemd160. * self.block\_chain\_db – עצם מסוג BlockChainDB המכיל את הבלוקצ'יין שהארנק משתמש בוא * self.unspent\_outputs – רשימת הoutputs הלא מבוזבזים. משמשים על מנת לחשב את מאזן הכסף שיש לארנק * self.balance – מאזן הכסף שיש לארנק * self.transactions – כל העסקאות שהעבירו כסף לארנק או שהעביר בהם כסף לארנק אחר * self.logger – עצם מסוג logger בשביל הפעולות במחלקה |
| פעולות   * \_\_init\_\_ - הפעולה יוצרת את הארנק ואת העצם המורש. הערכים שהיא מקבלת הם המפתח הפרטי של הארנק, עצם הבלוקצ'יין והlogger. * new\_wallet – פעולה סטאטית המחזירה עצם מסוג ארנק הנוצר ממפתח ציבורי חדש. הפעולה מקבלת את הערכים של עצם הבלוקצ'יין ואת הlogger.   השימוש בערכים אלו הוא בשביל ליצור בלוק חדש ולא כשמוציאים אותו ממבנה הנתונים   * can\_unlock\_output – הפעולה מקבלת output ומחזירה אמת אם הארנק יכול להשתמש בוא ליצירת input, כלומר לבזבז אותו, ושקר אחרת. * is\_unspent\_output – הפעולה מקבלת את העסקה שהoutput נמצא בא ואת המספר הסידורי שלו בא ומחזירה אמת אם הoutput הוא לא מבוזבז ושקר אחרת * update\_balance – הפעולה מעדכנת את המאזן הכספי של הארנק, בהתאם לשינויים שנעשו בבלוקצ'יין (אם נעשו). * create\_transaction – הפעולה מקבלת כמות כסף לשלוח ואת כתובת הארנק שלוח לה. העסקה מחזירה אמת אם יש לארנק מספיק כסף לעסקה ואת העסקה עצמה. בנוסף לכך, אם העסקה תקינה היא מוסיפה אותה למאגר העסקאות בבלוקצ'יין שלה. אחרת היא מחזירה שקר. * update\_unspent\_outputs – הפעולה מעדכנת את הoutputs הלא מבוזבזים של הארנק, בהתאם לשינויים בבלוקצ'יין (אם נעשו) * update\_transactions– הפעולה מעדכנת את רשימת העסקאות הקשורות בארנק בהתאם לשינויים בבלוקצ'יין (אם נעשו). * find\_address – פעולה סטאטית המחשבת את הכתובת של המחזיק במפתח הציבורי שהיא מקבלת ומחזירה אותה. * equals\_public\_key – הפעולה מקבלת מפתח ציבורי ומחזירה אמת אם הוא זהה (מתאים לאותו מפתח פרטי) למפתח הציבורי של הארנק, ושקר אחרת. |

|  |
| --- |
| NodeServer |
| תכונות   * self.server\_socket – עצם מסוג socket שמשמש את השרת (ראה ספרייה socket) * self.logger – עצם מסוג logger המשמש את הפעולות במחלקה * self.recieved\_messages – רשימת ההודעות שהשרת קיבל והם עוד לא טופלו * self.to\_close – דגל בוליאני המודי האם להפסיק את פעולת השרת * self.readable – רשימת הsockets שמוכנים לקריאה * self.exceptional – רשימת הsockets שאראה שגיאה בקשר איתם. |
| פעולות   * \_\_init\_\_ - הפעולה יוצרת את העצם ואת השרת. הפעולה מקבלת את הlogger. * run – פעולה שמריצה את השרת עד שקופץ הדגל של הסגירה. על מנת תפקוד תקין של התוכנית, הפעולה צריכה לרוץ בthread נפרד מהthread הראשי. * handle\_exception\_sockets – הפעולה מטפלת בsockets שאראה בהם שגיאה. * handle\_read\_sockets – הפעולה מטפלת בsockets שמוכנים לקריאה. * extract\_pack\_length – פעולה סטאטית המקבלת socket של לקוח ומקבלת ממנו את האורך של ההודעה שהוא רוצה להעביר בהתאם לפרוטוקול. הפעולה מחזירה אורך זה. * receive\_message – פעולה סטאטית המקבלת socket של לקוח ואת האורך של ההודעה שהוא אמור לקבל ממנו. הפעולה מקבלת ממנו את ההודעה בהתאם לאורכה ומחזירה אותה. * disconnect\_connection – הפעולה מקבלת socket של לקוח ומנתקת אתו באופן תקין את הקשר. * get\_received\_messages– הפעולה מחזירה רשימה של ההודעות שקיבל השרת והם עוד לא טופלו. * remove\_message – הפעולה מקבלת הודעה ומוציאה אותה מרשימת ההודעות שקובלו בשרת. |

|  |
| --- |
| NodeClient |
| תכונות   * self.logger – עצם מסוג logger המשמש את הפעולות במחלקה * self.known\_nodes – רשימת הכתובות הידועות שמחוברות (כתובות ip) * self.client\_socket – עצם מסוג socket המשמש את הלקוח * self.lock – מנעול מthreading המונע מלשנות את הרשימה של הכתובות הידועות ובאותו זמן גם לשלוח הודעות לכתובות האלו |
| פעולות   * \_\_init\_\_ - הפעולה יוצרת את העצם ואת הלקוח. הפעולה מקבלת את הlogger, ורשימה של כתובות ידועות * self.send\_to\_all – הפעולה מקבלת הודעה ושולחת אותה לכל הכתובות הידועות שמחוברות * self.send – הפונקציה מקבלת הודעה וכתובת לשלוח אליה והפונקציה שולחת את ההודעה אליה * check\_connections – הפונקציה בודקת שהכתובות הידועות מחוברות ואם לא מוציאה אותם מהרשימה |

|  |
| --- |
| Node |
| תכונות   * self.logger – עצם מסוג logger המשמש את הפעולות במחלקה * self.block\_chain\_db – עצם מסוג BlockChainDB שמשמש את המחלקה * self.address– כתובת הip הפנימית של הnode * self.server – עצם מסוג NodeServer האחראי על השרת * self.known\_nodes\_db – עצם המאפשר גישה למבנה הנתונים של הכתובות הידועות * self.known\_nodes- רשימה של כתובות של nodes מחוברות * self.client – עצם מסוג NodeClient האחראי על הלקוח * self.msg\_handler – עצם האחראי על לשנות עצמים של הודעות למחרוזת שאפשר לשלוח ולהפך * self.server\_thread – הthread שמריץ את הserver |
| פעולות   * \_\_init\_\_ - הפעולה יוצרת את הnode. הפעולה מקבלת את הlogger, ואת הבלוקצ'יין * update\_chain – הפעולה מעדכנת את הבלוקצ'יין לפי הבלוקצ'יין הארוך ביותר ברשת * find\_longest\_chain – הפעולה מקבלת את ההודעות שהשרת קיבל ומחזירה מהם את האורך הכי גדול של הבלוקצ'יין ברשת ואת המחזיקים בשרשרת זאת. * get\_longest\_chain – הפעולה מקבלת את האורך של השרשרת הארוכה ביותר ואת הכתובות של המחזיקים בה ומורידה מהאינטרנט אותה * extract\_blocks\_hashes – פעולה סטאטית המחלצת מהודעות הinv את רשימת קודי הhash שמרבית הnodes מסכימה עליהם ומחזירה אותם. * download\_blocks – הפעולה מורידה מהאינטרנט את הבלוקים לפי ערכי הhash שהיא מקבלת שלהם ומחזירה אותם. * handle\_mesages – הפעולה מטפלת בהודעות שקיבל השרת. * handle\_version– הפעולה מטפלת בהודעה מסוג version שהיא מקבלת * handle\_inv – הפעולה מטפלת בהודעה מסוג inv שהיא מקבלת * handle\_get\_data – הפעולה מטפלת בהודעה מסוג get data שהיא מקבלת * handle\_block\_message – הפעולה מטפלת בהודעה מסוג block שהיא מקבלת * handle\_transaction\_message – הפעולה מטפלת בהודעה מסוג transaction שהיא מקבלת * handle\_get\_addresses\_code – הפעולה מטפלת בהודעה מסוג get\_addresses שהיא מקבלת * handle\_addresses\_code – הפעולה מטפלת בהודעה מסוג addresses\_code * find\_connections – הפעולה מוצאת רשימת nodes שאפשר להתחבר אליהם * verify\_transaction – הפעולה מאמתת את העסקה שהיא מקבלת ומחזירה אמת אם * is\_unspent\_output – הפעולה מקבלת עסקה ואת האינדקס של הoutput בה ומחזירה אמת אם הוא output לא מבוזבז ושקר אחרת * verify\_proofs – הפעולה מקבלת עסקה ומאמתת את ההוכחות בה. מחזירה אמת אם כל ההוכחות חוקיות ושקר אחרת * verify\_amount – הפעולה מאמתת את התשלום עצמו (אם יש מספיק כסף) בעסקה שהיא מקבלת. מחזירה אמת אם הוא חוקי ושקר אחרת * check\_if\_inside – הפעולה מקבלת hash קוד של עסקה ומחזירה אמת אם היא נמצאת בבלוקצ'יין ושקר אחרת |

|  |
| --- |
| GuiWindow |
| תכונות   * self.win\_dict – מילון מחלקות החלונות השונים במערכת * self.wallet – ארנק המשתמש * self.top – חלון הTk שעליו יוצרים את הframes |
| פעולות   * \_\_init\_\_ - הפעולה יוצרת את החלון. הפעולה מקבלת את חלון ה,Tk את מילון מחלקות החלונות ואת ארנק המשתמש. * set\_window\_geometry – הפעולה מחשבת את הגודל והמיקום של המסך בהתאם לגודל המחשב * transaction\_to\_string – הפעולה מקבלת עסקה והופכת אותה למחרוזת שאפשר להציג על החלון. הפעולה מחזירה את המחרוזת |

|  |
| --- |
| WalletMainWindow |
| המחלקה יורשת מעצם GuiWindow  תכונות   * הframes השונים שמחלקים את החלון, הכפתורים, הlabels והטקסטים שבהם. |
| פעולות   * \_\_init\_\_ - הפעולה יוצרת את העצם המורש ואת הframes בחלון. הפעולה מקבלת את חלון הTk, את מילון המחלקות ואת ארנק המשתמש. * פעולות שיוצרות את הframes השונים ואת מה שבתוכם * pressed\_back – הפעולה מטפלת בלחיצה על הכפתור שמעביר אחורה למסך הראשי * pressed\_send – הפעולה מטפלת ששולח את העסקה |

|  |
| --- |
| SendWindow |
| המחלקה יורשת מעצם GuiWindow  תכונות   * הframes השונים שמחלקים את החלון, הכפתורים והlabels שבהם. * העסקה שנוצרת בידי המשתמש |
| פעולות   * \_\_init\_\_ - הפעולה יוצרת את העצם המורש ואת הframes בחלון. * פעולות שיוצרות את הframes השונים ואת מה שבתוכם * pressed\_send – הפעולה מטפלת בלחיצה על הכפתור שמעביר למסך השליחה * pressed\_transactions – הפעולה מטפלת בלחיצה על הכפתור שמראה את כל העסקאות של הארנק * show\_current\_balance – הפעולה מראה על החלון את המאזן הכספי הנוכחי של הארנק * show\_last\_transactions – הפעולה מראה את העסקאות האחרונות שנעשו עם הארנק * show\_address – הפעולה מראה את הכתובת של הארנק |

|  |
| --- |
| TransactionsWindow |
| המחלקה יורשת מעצם GuiWindow  תכונות   * הframes השונים שמחלקים את החלון, הכפתורים, הlabels והטקסט שבהם. |
| פעולות   * \_\_init\_\_ - הפעולה יוצרת את העצם המורש ואת הframes בחלון. הפעולה מקבלת את חלון הTk, את מילון המחלקות ואת ארנק המשתמש. * פעולות שיוצרות את הframes השונים ואת מה שבתוכם * pressed\_back – הפעולה מטפלת בלחיצה על הכפתור שמעביר אחורה למסך הראשי * show\_transactions – הפעולה מראה על המסך את כל העסקאות שנעשו עם הארנק. |

|  |
| --- |
| UiRunner |
| תכונות   * self.loggers – רשימת עצמים מסוג logger לשימוש במחלקות השונות * self.block\_chain\_db – הבלוקצ'יין של המשתמש * self.root – מסך הTk שרץ * self.current\_window – עצם החלון הנוכחי שמציג * self.win\_dict – מילון מחלקות השונות * self.wallet – הארנק של המשתמש * self.node – עצם הnode שמשמש את המשתמש |
| פעולות   * \_\_init\_\_ - הפעולה יוצרת את העצם, את מילון החלונות, את הארנק והnode * run - מריצה את הGUI |

|  |
| --- |
| Miner |
| תכונות   * self.wallet – הארנק של הכורה * self.node – עצם הNode של הכורה המקשר בינו לבין הnodes האחרים * self.failed – דגל שקופץ אם כורה אחר הקדים את הכורה |
| פעולות   * \_\_init\_\_ - הפעולה יוצרת הכורה ומתחילה בתהליך הכרייה * mine – הפעולה מנסה לכרות את הבלוקים * handle\_received\_messages - הפעולה מטפלת בהודעות המתאימות שהשרת מקבל |

תרשים Use Case

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

2.3 דיון בנושא העיצוב הנבחר

בחרתי בחלוקת רכיבים זאת משום שהיא מאפשרת חילוק נכון ונוח למחלקות, ובכך לשמור את הבלוקצ'יין בצורה הטובה ביותר. בנוסף לכך הפרדתי בתוכנית את הרכיב של הnode והארנק על מנת לאפשר לכל רכיב לפעול בלי קשר לאחר, ובכך לנהל אותם בצורה הטובה ביותר. הרכיבים כוללים ביניהם עצם בלוקצ'יין משותף על מנת לאפשר לכל רכיב לשנות את השדות שהוא צריך וזה ישפיע על כל המערכת. לדוגמא, הnode מוסיף בלוק לשרשרת, לכן הארנק צריך להיות מודע לבלוק הזה כי הוא יכול לכלול עסקאות שקשורות בוא.

את הlogging עשיתי כך שיש קבצי log שונים למערכות השונות. יכולתי ליצור קובץ אחד אבל העדפתי על מנת לראות בצורה יותר נוחה את פעולות של כל רכיב, לחלק לקבצים שונים.

את תהליך מציאת הknown peers אפשר לעשות בדרכים שונות: שרתים ידועים שעוזרים למצוא, שידור לכל הרשת את ההודעת get\_addresses או לשמור מבנה נתונים של כל הnodes הידועים הכוללnodes מארחים, כלומר ידוע שהם רצים כל הזמן או מרבית מהזמן והם יכולים לעזור למצוא כתובות. לאחר שחקרתי על איך bitcoin עושה זאת ועל האפשרויות השונות. החלטתי להשתמש בפתרון השלישי משום שהמימוש שלו דיי נוח ולא מסורבל וגם הוא דיי קרוב קרוב לרשת p2p טהורה.

את מבני הנתונים (מלבד שמירת קובץ הארנק) החלטתי ליישם בsql ולשמור אותם בטבלאות משום שזה מסדר את הנתונים בצורה הכי נוחה לגישה. קובץ המפתח הפרטי אינו ממומש בsql משום שזה עצם אחד שאין כל היגיון לשמור בטבלאות. במקום זאת, הוא נשמר בעזרת pickle והוא ניתן להפוך למחרוזת בקלות בעזרת המימוש שלו על ידי pycrypto.

כל הקוד הוא בשפת python משום שמימוש מבני נתונים ותכנות מומחה עצמים הוא מאוד נוח בשפה הזאת, וגם מימוש הרשתות.

1. עיצוב נתונים ופרוטוקולים

מבני נתונים

מבני הנתונים השומרים את כתובות הnodes ואת הבלוקצ'יין כתובים בשפת sql ונשמרים בטבלאות, למעט קובץ המפתח הפרטי שנשמר בקובץ בעזרת pickle.

הבלוקצ'יין

שרשרת הבלוקים נשמרת במערכת על ידי ארבעה טבלאות: טבלת הבלוקים, טבלת העסקאות, טבלת הinputs וטבלת הoutputs.

העמודות בטבלת הבלוקים הינם (ניתן לראות את המשמעות של כל תכונה בתיעוד המחלקות):

* number – int
* nonce – int
* prev – real
* difficulty – int
* time\_stamp - real

העמודות בטבלת העסקאות הינם:

* transaction\_id – int. המספר הסידורי של העסקה בטבלה (מתחיל מ0)
* block\_number– int. המספר הסידורי של הבלוק בטבלת הבלוקים שהעסקה שייכת אליו

העמודות בטבלת הinputs הינם (ניתן לראות את המשמעות של כל תכונה בתיעוד המחלקות):

* transaction\_id – string
* output\_index – int
* proof – string
* transaction\_number – string. קוד הhash של העסקה שהinput שייך אליה

העמודות בטבלת הoutputs הינם (ניתן לראות את המשמעות של כל תכונה בתיעוד המחלקות):

* value – float
* address – string
* transaction\_number – string. קוד הhash של העסקה שהoutput שייך אליה

כתובות הnodes

הכתובות מחולקות לשני מבני נתונים. אחד כולל את כל ה nodesואחר את הnodes המארחים. התוכנית חייבת את הקובץ של הnodes המארחים על מנת להתחבר בהתחלה למערכת. קובץ זה הוא קבוע.

העמודות במבנה נתונים זה הם קבועות והן כוללות עמודה של address המכילהstring. בעמודה זאת יש את הכתובת. עמודה של id המכילה int המציין את המספר הסידורי של הכתובת (מתחיל מ0).

קובץ המפתח הפרטי

קובץ זה הוא הקובץ שהמשתמש חייב לשמור על מנת להיכנס לארנק שלו. הקובץ שומר את המפתח הפרטי בקוד ascii.

פרוטוקול התקשורת (הרחבה על התקשורת במסמך החקר)

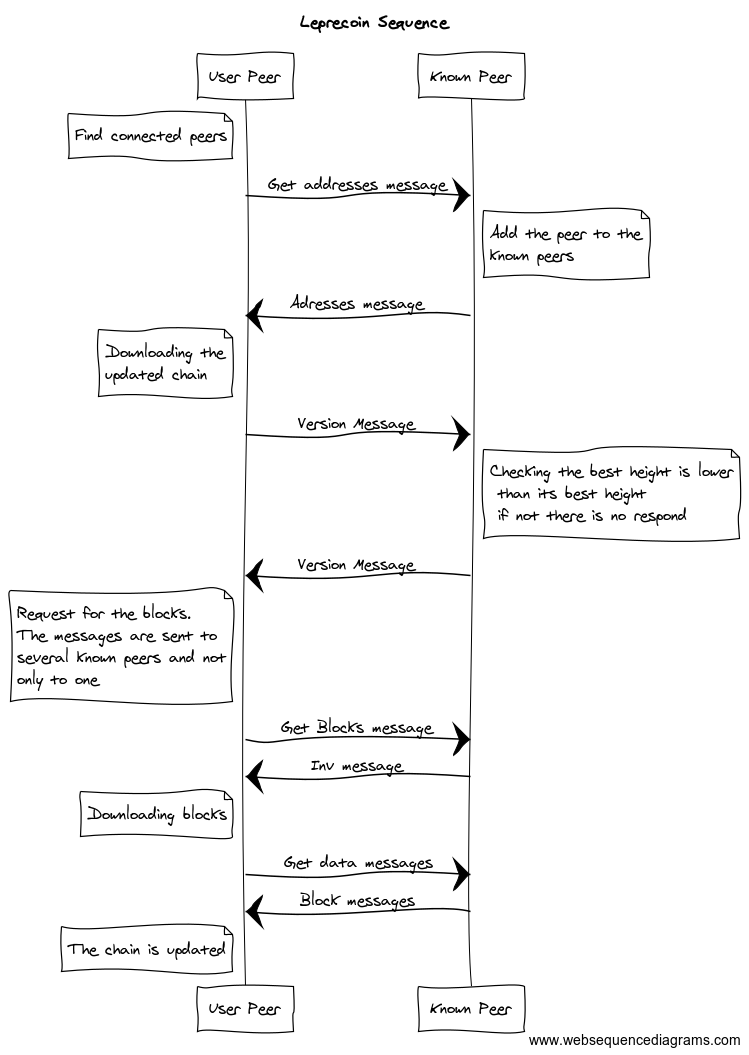
פרוטוקול התקשורת מורכב מההודעות הבאות:

* Version- כאשר node מתחבר למערכת מחדש הוא שולח את ההודעה הזאת והוא בעצם מעדכן את הרשת במצב הבלוקצ'יין שלו. ההודעה מכילה את האורך של הבלוקצ'יין שלו. כתגובה הpeers האחרים מחזירים לו הודעת version אם מצב הבלוקצ'יין שלו לא מעודכן ממצב האחרים הוא צריך להוריד את הבלוקצ'יין המעודכן מהרשת.
* Getaddresses – בקשה לכתובות הפעילות המוכרות לnode שמקבל אותה
* Addr- הודעה זאת מכילה רשימת כתובות שהnode הנוכחי מכיר.
* Getblocks- בקשה להראות אילו בלוקים יש לך מהבלוק האחרון שיש לי. בתשובה תהיה בהודעת inv .
* Inv- הודעה זאת מראה אילו עסקאות או בלוקים יש לשולח. ההודעה מכילה את הסוג של המידע שבתוכה- בלוק או עסקה ואת הhashes של הבלוקים או העסקאות.
* Getdata- הודעה זאת היא בקשה לעסקה מסוימת או בלוק מסוים. ההודעה מכילה את סוג המידע המבוקש ואת הhash שלו.
* Transaction- הודעה זאת היא תגובה לgetdata של עסקה והיא מכילה מחרוזת שממנה יוכל המקבל להרכיב את העצם של העסקה.
* Block- הודעה זאת היא תגובה לgetdata של בלוק והיא מכילה מחרוזת שממנה יוכל המקבל להרכיב את העצם של הבלוק.

תרשים sequence diagram

מאחר שכל מחשב מתנהג בתקשורת מסוג זאת על פי אותם חוקים אז את התרשים ניתן להפוך וכל צד מתנהג בדיוק כמו האחר:

תרשים של התחברות חדשה המצריכה רשימה של כתובות ואת הבלוקצ'יין העדכני:



תרשים של ההודעות שיכולות להיות אחרי שהnode/peer מחובר עם השרשרת העדכנית:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תרשים של הכורה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

כאשר כורה אחר הקדים את הכורה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

\* בתרשימים הוצגו תקשורת בין שני רכיבים בלבד. בפועל ההודעות נשלחות לפעמים (בהתאם לצורך), לכמה nodes. לדוגמא, כאשר מורידים את הבלוקצ'יין מהnodes ההורדה נעשית מכמה nodes ומכל אחד מבקשים בלוק אחר.

\*בתרשים כתוב peer אך כפי שנאמר בתקשורת שני המושגים node וpeer מתייחסים לאותו דבר.

4. ממשק משתמש

כאן יופיע פירוט ממשקי המשתמשים וכן תופיע הסקיצה עבורם

בסעיף זה יש לפרט את הפונקציונליות של המערכת כפי שהיא מתבטאת עבור משתמש חיצוני. במידה ויש סוגים שונים של משתמשים, יש להתייחס לכולם בהתאם - אילו רכיבים או נתונים רלוונטיים עבורם, וכיצד הם מתקשרים איתם.  
  
בנוסף, יש לצרף המחשות ויזואליות של המסכים השונים, ולהסביר את התוכן שלהם (למשל מה התפקיד של כל שדה או כפתור, מה קשור/תלוי במה - למשל כפתור שמכובה בהתאם לתנאים מסוימים במערכת וכן הלאה) והקשרים ביניהם (איזה מסך מוביל לאיזה מסך ובאילו מקרים)  
  
דוגמה (לא מלאה):  


5. נספחים

**כל דבר שהייתם רוצים להוסיף בנוגע לעיצוב המערכת**