

פרויקט גמר בסייבר

מסמך עיצוב

**שם התלמיד: עומר פרי**

**ת"ז: 318444528**

**בית הספר: אורט גוטמן**

**שמות המורים: מיכאל וגד**

**מועד הגשה: 02/2016**

**מבוא**

כאשר עובדים על פרויקט במחשב, מכל סוג, נתקלים לעיתים רבות בבעיות שונות. הבעיות הללו עוברות מחוסר גיבוי במידה ונהרס המחשב, ועד שינוי קטן שנרצה לבטל. מי מאתנו לא נתקל לפעמים בבעיה מסוג זה? לפתע הפרויקט שעבדת עליו כל כך הרבה זמן מפסיק לעבוד ואתה לא מצליח לברר מה הבעיה, אך לצערך לא שמרת את הגרסה האחרונה אשר עבדה. כמו כן, לפעמים נרצה אולי לעבוד על הפרויקט שלנו מעמדת עבודה שונה אך לא נוכל לעשות זאת בשל העובדה שלא שמרנו אותו על דיסק-און-קי, או שזה אינו נמצא בידינו. נוסף על כך, לעיתים נרצה לראות מתי שינינו משהו ספציפי בפרויקט, או שלא נזכור מתי ביצענו את השינוי ונרצה דרך נוחה לגלות זאת.  
  
המוטיבציה לפיתוח התוכנה מורכבת מכמה חלקים. ראשית, לעולם לא נגעתי בתחום הזה וזה נראה לי מאוד מעניין. אמנם קיים כבר כלי אשר נותן לנהל פרויקט בצורה מסודרת ולנהל גרסאות שונות של קבצים, אבל לא קיים כלי כזה שהוא נוח לשימוש וכל בן אדם (גם כזה שלא מבין כלום במחשבים) יוכל להשתמש בו. כמו כן, לא קיים כלי אשר מאפשר ויזואליזציה של הגרסאות. משתי הסיבות שצוינו לעיל הפרויקט הזה יהיה מאתגר ומעניין שכן עוד לא קיים דבר כזה, לכן אאלץ לחשוב על פתרונות לבעיות שעוד לא נתקלו בהן ודרכים לעשות דברים (כמו לדוגמה, איך להפוך את הכלי לאופטימלי) בעצמי מבלי שאוכל להיעזר באחרים.

התוכנה אמורה לאפשר ניהול נוח של פרויקט תוך כדי תמיכה בויזואליזציה של היסטוריית הקבצים והפרויקט ואף נתינת גיבוי של הפרויקט על שרת חיצוני.

**תיאור המוצר**

המטרה המרכזית של המוצר היא לאפשר ניהול נוח ככל האפשר של פרויקטים על המחשב. זאת על ידי גיבוי הפרויקט (ומתן טעינתו מכל מקום), שמירת גרסאות קודמות של קבצים ואף ויזואליזציה של היסטוריית הפרויקט והקבצים.

התרחישים שהמערכת צריכה להתמודד איתם הם:

* רישום והתחברות של משתמש. (ניהול מאגר מידע הכולל את המשתמשים ואת הפרויקטים שלהם)
* משתמש פותח פרויקט, מוסיף / מוחק קבצים, מתחיל גרסה חדשה של קובץ קיים, עורך גרסה ישנה של קובץ קיים וכו'...
* משתמש מבקש להוריד את הפרויקט שלו מהשרת אשר מגבה את הפרויקט בכל שלב.
* משתמש רוצה לראות בצורה ויזואלית את היסטוריית הפרויקט שלו / את ההיסטוריה של אחד הקבצים בפרויקט.

**אילוצים ודרישות**

הבעיות שהמערכת צריכה לדעת להתמודד איתן הן:

1. המערכת צריכה לשמור את הפרויקט לשרת ואף במחשב הלוקאלי מבלי להפריע לעבודת המשתמש.
2. המערכת צריכה להיות נוחה ופשוטה ככל האפשר.
3. מערכת הויזואליזציה צריכה לעבוד בצורה חלקה כך שהמשתמש לא יבחין בתקיעות בזמן השימוש בה.

היא נותנת להן מענה בכך ש:

1. המערכת תשמור כל זמן קצוב / מספר מסוים של שינויים את הקובץ במחשב של המשתמש ובסוף השימוש / בעת בקשת המשתמש היא תעלה את הפרויקט לשרת.
2. למערכת יהיה ממשק מאוד פשוט ומובן.
3. המערכת תטען חלקים מהקובץ בצורה חכמה כך שהשימוש יהיה חלק ככל האפשר.

**תיחום הפרויקט**

הפרויקט משלב את תחום גיבוי הקבצים, ויזואליזציה ותקשורת בין המחשבים.

הפרויקט יעשה שימוש במודולים שיאפשרו לו:

* תקשורת בין מחשבים
* גיבוי וגישה לקבצים
* GUI

**סביבת העבודה**

מירב הפרויקט ייכתב ב Python (התקשורת בין הלקוח לשרת, הטיפול בקבצים) ואילו ה GUI ייכתב בשפת C#.

**ארכיטקטורת המערכת**

**רשימת מודלים:**

צד שרת:

* void databaseExists()
  + הפונקציה בודקת האם קיים בשרת מאגר נתונים.
  + הפונקציה מחזירה ערך בוליאני בהתאם.
* databaseInit() void
  + הפונקציה יוצרת מאגר נתונים בשרת.
* socket createServer()
  + הפונקציה יוצרת שרת שאליו יתחברו הלקוחות.
  + הפונקציה מחזירה נקודת קצה שדרכה יישלחו נתונים לשרת.
* bool isLogged(socket Client)
  + הפונקציה בודקת האם המשתמש מחובר.
  + הפונקציה מחזירה ערך בוליאני בהתאם.
* int Login(string username, string password)
  + הפונקציה מנסה לבצע התחברות של לקוח עם הפרטים שהוא סיפק.
  + הפונקציה מחזירה קוד בהתאם לתוצאות הניסיון: שם משתמש לא מזוהה, סיסמה לא נכונה או התחברות מוצלחת.
* int signUp(string username, string password)
  + הפונקציה מנסה לבצע הרשמה של לקוח עם הפרטים שהוא סיפק.
  + הפונקציה מחזירה קוד בהתאם לתוצאות הניסיון: שם משתמש תפוס, או הרשמה מוצלחת.
* Void sendFileToClient(string path)
  + הפונקציה שולחת למשתמש קובץ מהשרת.
* Void downloadFileFromClient(string path)
  + הפונקציה מקבלת קובץ מן המשתמש.

צד לקוח:

* socket connectToServer()
  + הפונקציה מחברת את הלקוח אל השרת.
  + הפונקציה מחזירה נקודת קצה שדרכה יישלחו נתונים אל השרת.
* connectToGUI() socket
  + הפונקציה מחברת את הלקוח אל ה GUI
  + הפונקציה מחזירה נקודת קצה שדרכה יישלחו נתונים אל הGUI.
* isInit() bool
  + הפונקציה בודקת האם זוהי הפעם הראשונה שהמשתמש נכנס לתוכנה במחשב שעליו הוא עובד.
  + הפונקציה מחזירה ערך בוליאני בהתאם.
* Initialize() void
  + הפונקציה יוצרת את תיקיית התוכנה במחשב הנוכחי.
* void localSaveVersion(string path)
  + שומר את הגרסא הנוכחית של הקובץ במחשב הלוקאלי.
* void serverSaveVersion(string path)
  + שומר את הגרסא הנוכחית של הקובץ בשרת.
* File downloadFileFromServer(string path)
  + מבקש מהשרת קובץ ספציפי.
  + מחזיר אובייקט של הקובץ שהורד.
* void openFile(string path)
  + פותח את הקובץ במיקום הנבחר דרך ה explorer.
* bool inSync()
  + בודק האם הגיבוי הלוקאלי תואם את הגיבוי שעל השרת.
  + מחזיר ערך בוליאני בהתאם.
* Void Sync()
  + מסנכרן בין השרת לבין הגיבוי הלוקאלי.
* Project addProject(string name)
  + יוצר פרוייקט חדש בשם הנבחר.
  + מחזיר אובייקט של הפרוייקט.
* Void viewHistory(string path)
  + שולחת בקשה לGUI להציג את היסטוריית הקובץ.
* String hash(string text)
  + מגבב את הטקסט שמעבירים לפונקציה.
  + מחזיר את הטקסט המגובב.

**מחלקות:**

Class File(Object)

* String path
* void rename(string name)
  + הפונקציה משנה את שם הקובץ.
* Void delete()
  + הפונקציה מוחקת את הקובץ.

Class oFile(File)

* String path
* Dictionary data
* Void load()
  + משמש לעדכון המילון מהקובץ.
* Void update()
  + משמש לעדכון הקובץ ע"י המילון.
* Void set(tuple/Dictionary data)
  + משמש להשמת ערכים בקובץ
* Void remove(key)
  + מוחק ערך ספציפי מהקובץ.
* String read(key)
  + מחזיר את ערך המפתח בקובץ.
* List of strings readall()
  + מחזירה את כל הערכים בקובץ.
* List of strings readkeys()
  + מחזירה את כל המפתחות בקובץ.

Class Directory(Object)

* String path
* Void rename(string name)
  + משנה את שם התיקייה.
* Void remove(string name)
  + מוחק את התיקייה.

Class Utilities(Object)

* Static File create\_file(path)
  + יוצר קובץ חדש.
  + מחזיר אובייקט של הקובץ.
* Static File open\_file(path)
  + מחזיר אובייקט של הקובץ.
* Static Directory create\_dir(path)
  + יוצר תיקייה חדשה.
  + מחזיר אובייקט של התיקייה.
* Static Directory open\_dir(path)
  + מחזיר אובייקט של התיקייה.
* Static bool path\_exists(path)
  + בודק האם הנתיב קיים.
  + מחזיר ערך בוליאני בהתאם.
* Static open\_o\_file(path)
  + מחזיר אובייקט של הקובץ מסוג oFile.
* Static create\_o\_file(path)
  + יוצר קובץ חדש.
  + מחזיר אובייקט של הקובץ מסוג oFile.

Class Project(Object)

* String name
* File Config
* List of files' names
* String getName()
  + הפונקציה מחזירה את שם הפורייקט.
* List of strings getFilesNames()
  + הפונקציה מחזירה רשימה של שמות הקבצים.
* List of string getFilesPaths()
  + הפונקציה מחזירה רשימה של מיקומי הקבצים.
* void renameProject(string name)
  + הפונקציה משנה את שם הפרוייקט.
* void deleteProject()
  + הפונקצייה מוחקת את הפרוייקט.
* void addFile(string path)
  + הפונקציה מוסיפה קובץ לפרוייקט.
* void removeFile(string path)
* void renameFile(string path, string name)
  + הפונקציה משנה את שם הקובץ.
* File getFile(string path)
  + מחזיר אובייקט של הקובץ.
* File getHistoryList(string path)
  + מחזיר רשימה של היסטוריית הקובץ.
* void saveFileVersion(string path)
  + שומר את הגרסא הנוכחית של הקובץ (לוקאלית ובשרת).

**קשר בין המתודות:**

* Client Connects:
  + Logged In
    - Start Program
      * Check if initialization is needed
        + Initialize (Create .VersionVisualizer)
      * Check if Sync is needed
        + Sync
      * Choose Project
        + Option to view and open files

Get a list of the project's files

Open a file (find path in Config.ini)

* + - * + Option to view older versions of a file

Get a list of the file's older versions

Fetch an older version and view it

* + - * + Option to create a new version of a file

Save a version in local machine

Save a version in the server

* + - * + Option to add a new file to the project
      * New Project
        + Select Files
  + Logged out
    - Register
    - Login

**מבנה בסיס הנתונים:**

Server

Name of User

* Config
* Name of Project
  + Metadata (files' Hierarchy)
  + Folder for each file
    - Old Versions
  + Latest Version of each file

Client

.VersionVisualizer

* Users
  + File for each User
* Folder for each User
  + Config (Contains path for each file)
  + Folder for each file
    - Old versions of files

**התלבטויות + אופציות להרחבות**

* Compress files
* Branching
* Check-in
* Other Formats
* Renamed Files (Older Versions)
* Deleted Files (Older Versions)
* MD5 | SHA2
* Encryption Server | Client
* Offline Login Access
* Older Versions of the project itself
* Names of file's versions (add comments)