**פיתוח תוכנה מתקדם 2**

**אבן דרך 1**

|  |  |  |
| --- | --- | --- |
| שם מלא | תעודת זהות | אימייל |
| הדר בסון | 314647462 | hadarbsn@gmail.com |
| נועם לושי | 31232207 | noamlushi@gmail.com |
| אנסטסיה טרנובסקי | 324561802 | nastyaternovski@gmail.com |
| נועם מאיר | 206379216 | noam220599@gmail.com |
| דקל רומנו | 206137895 | dekelr26@gmail.com |
| בן ניזרי | 318431012 | bennizri22@gmail.com |
| דפנה פורת | 318797685 | dafna.porat12@gmail.com |

בפורמט זה נסקור את שלושת היישויות הראשיות בפרויקט ונפרט אודות כל יישות:

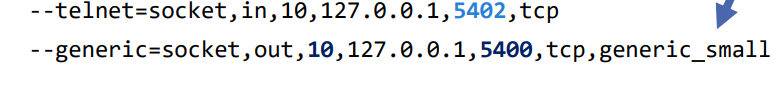
**Agent**

ה-agent מותקן על כל מטוס כלומר על כל יחידת קצה שבו נריץ מופע של סימולטור הטיסה ולו מס' תפקידים חשובים:

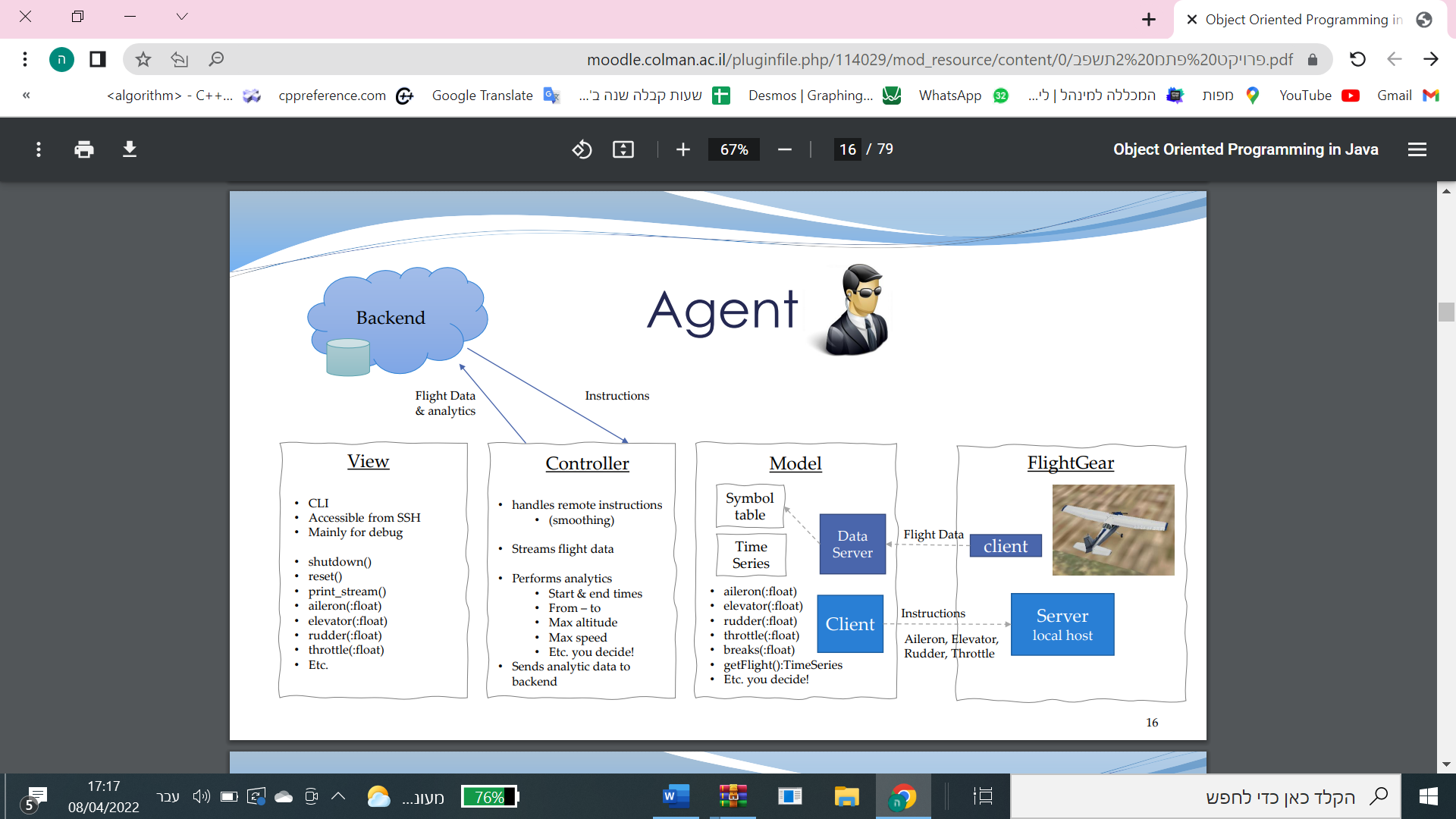
* איסוף מידע: באמצעות ה- agentנדלה אינפורמציה אודות פעילות המטוס כגון: מתי המטוס המריא? מתי נחת? מאיפה הטיסה יצאה ומה יעדה? מה היה הגבוה המקסימלי במהלך הטיסה? ועוד מגוון שאלות. את כלל הנתונים הללו ה- agent ידלה מסימולטור הטיסה וכל זאת בעזרת קובץ XML שייעדו להגדיר לסימולטור מהם הנתונים שמעניינים אותנו. (קובץ שאותו אנו מקבלים מסגל הקורס).
* הזרמת מידע: ה- agentמזרים את המידע אודות הטיסה ל-backend על מנת שמירת המידע בשרת DB ייעודי.
* העברת פקודות: ה- agentמקבל פקודות מה- backendודואג להעביר אותן לסימולטור הטיסה על מנת שפקודות אלה יבוצעו בזמן אמת וישנו את נתוני הטיסה.

במהלך העבודה אנו סבורים כי נחשף לעוד תפקידים חושבים של ישות זאת.

ה- agentמתקשר עם ה-FlightGear באמצעות שני ערוצי תקשורת- כשרת וכלקוח ע"י שני threads נפרדים הרצים במקביל. בסימולטור הטיסה עלינו להגדיר את ערוצי התקשורת ע"י הפקודות הללו במסך ההגדרות של הסימולטור:



ה- agentבנוי בארכיטקטורת MVC בו קיימים שלושת החלקים: Model,Controller,View כמפורט בשקף:



Flightgear-

תוכנה חיצונית המדמה טיסה על כלל מאפייניו. התוכנה תשמש אותנו בשני אופנים:

Client- מעביר נתונים אודות טיסה ל-model.

Server- השרת יקבל פקודות מה- modelשלפיהם נוכל לשלוט על מהלך הטיסה.

Model-

שכבה זו כוללת בתוכה את הפונקציונליות והפקודות לשליטת הטיסה, שכבה זו מתקשרת עם סימולטור הטיסה באמצעות שני Threads נפרדים- אחד כשרת ואחד כלקוח:

Client- מתחבר לצד השרת ב- Flightgearובאמצעותו נזריק פקודות אשר ישפיעו על מהלך הטיסה.

Server- יקבל דגימות מידע מהסימולטור אודות נתוני הטיסה בקצב מסויים לשנייה אותו נגדיר. נתוני הטיסה יוזרמו בצורת מילים מופרדות בפסיק אשר ישמורו ב-Symbol table. כמו כן, מהנתונים הללו נוכל ליצור מופע של time series(אותו אובייקט שאפיינו בקורס פת"ם 1).

-View

שכבה אשר עיקרה להוות כלי לביצוע בדיקות debug עבור ה -.agentבאמצעותו נוכל לבצע בדיקת תקינות מעבר הנתונים באופן מקומי.

הכלי הנ"ל ימומש בצורת CLI שבו נוכל להריץ פונקציות של בדיקות ולקבל מידע כגון: הדפסת נתוני כלל הטיסה או עבור מאפיין ספציפי. נוכל לבצע restart או לכבות לגמרי את המערכת וכו'.

Controller-

שכבה אשר מתקשרת עםbackend וכמובן שמקשרת בין שכבות ה-model וה-view,

באמצעותה נוכל לקבל פקודות מה- backend שישפיעו על מהלך הטיסה, ה Controller- יעביר פקודת אלה למודל שיעביר אותן בתפקידו לסימולטור ופקודות אלה יגולמו ויזואלית.

נוכל "להסטרים" את המידע כלומר להעבירו מסימולטור הטיסה ל .backend

בנוסף, נוכל לקבל אנליטיקות אודות טיסה שהסתיימה- כמו מתי הסתיימה? מתי התחילה?, גובה מקסימלי וכו'.

ל-controller תוגדר פונקצליות שבאמצעותה יוכל להחליט כיצד יקבל את המידע ואף יוכל "להחליקו" כלומר יחליט כיצד הוא מעביר או מקבל מידע כגון: גילום מס' נתוני טיסה כנתון אחד ,יחליט מה יהיה קצב העברת מידע הטיסה ל-backend ולהיפך וכו'.

**Backend**

הbackend הוא הגורם המתווך בין הfrontend שם נמצא הממשק שימוש של הלקוח ולבין הagent אשר נמצא בכל מטוס.

הbackend מורכב מ3 מרכיבים:

1) מסד נתונים מבוסס SQL אשר מקבל ומאחסן נתוני טיסה מהagent ומייצא נתוני טיסה לfrontend.

2) ארכיטקטורת MVC אשר מספקת שירותים של תיווך בין הfrontend לagent, בין הfrontend למסד נתונים ובין הagent למסד נתונים. שירותים אלה כוללים העברת נתוני טיסה, הוראות טיסה ושאילתות למסד נתונים.

3) שכבת ORM המתווכת בין הרמה של תכנות מכוון עצמים לרמה הראליציונית של המסד נתונים ומתרגמת אובייקטים לשאילתות.

ארכיטקטורת הMVC:

* View: משמש כCLI לצורך בדיקות, ובקרה על המשימות הפועלות בתוכנה.
* Controller: זו השכבה שמקבלת את הבקשות מforntend או מהagent. הcontroller פועל בצורה מקבילית, ומשתמש בטכנולוגיה של active object וthread pool אשר אותם נלמד בהמשך הקורס.
* Model: מספק את הגישה אל מסד הנתונים ומספק שירותים ואינטרקציות.

שירותים של הmodel:

אחסון ויצוא מידע מהמסד נתונים

* כאשר הסתיימה טיסה הagent פונה לbackend ומבקש ממנו לשמור את הנתונים בDB.
* על מנת להציג א כל הנתונים בfleet overview הfrontend שולח שאילתה לbackend והוא מתרגם אותה לשאילתת SQL וידלה את הנתונים הרצויים ויחזיר אותם לfrontend על מנת להציגם.

"הסטרמה" של מידע:

* מעביר נתונים, בקשות ושאילתות ומתווך בזמן אמת בין הfrontend לagent. לדוגמא בmonitoring- הbackend הוא זה שמעביר את הבקשות לagent: כאשר נרצה להציג נתונים של מטוס שעכשיו בטיסה הfrontend מעביר בקשה לbackend עבור אותו מטוס והbackend מבקש מהagent את הנתונים עבור אותו מטוס על מנת להחזירם לfrontend כדי להציגם. דוגמה נוספת היא בteleopration שם הfrontend מעביר פקודות שליטה על המטוס בזמן אמת והbackend מעביר את הפקודות האלו לagent על מנת להעביר אותם לסימולטור.
* גם לtime capsule הbackend מקבל ממנו שאילתות על טיסה מסוימת ויחזיר קובץ CSV על מנת שהtime capsule ישחזר את הטיסה.

Interpreter:

* בteleopration הbackend משמש כמפרש בין הקוד של הלקוח לבין מה שהagent מבין.

**Frontend**

במסגרת ישות זו- נבנה אפליקציית דסקטופ שתכיל 4 טאבים עיקריים:

**Fleet Overview**

יראה את המיקום של כל מטוס בצי המטוסים על מפת העולם בזמן אמת במסגרתו יהיה עדכון כל יחידת זמן (שנגדיר מראש) אודות המיקום של המטוסים בעזרת תקשורת עם ה-Backend,בנוסף כל מטוס יוצג על המסך עם כיוונו לאזימוט שהוא טס אליו.

-בלחיצה אחת על מטוס מסוים תוצג טבלה עם הנתונים הבאים : שם המטוס,כיוון הטיסה,גובה המטוס,מהירות המטוס וכו'.

- בלחיצה כפולה על המטוס נגיע לטאב הMonitoring אשר יציג נתונים נוספים אודות המטוס.

- נוכל להגדיל ולכווץ את תצוגת המפה כמו כן גם לטייל בה.

בנוסף, מתחת למפה של המטוסים יופיעו KPI– ים שונים בתצוגות Chart שונות:

Pie chart – אחוז המטוסים הפעילים מול הלא פעילים.

Bar chart :

1. מראה בצורה ממוינת כמה מיילים ימיים עשה כל מטוס במצטבר מתחילת החודש.

2. מראה בצורה ממוינת כמה מיילים ימיים עשו בממוצע כל מטוסי הצי בכל חודש מתחילת השנה.

Line chart – מראה את גודל הצי לפי זמן.

\*בהמשך העבודה נוסיף פיצ'רים נוספים.

**Monitoring**

יציג מדדים שונים בזמן אמת על מטוס בודד.

בחלק זה נראה את המידע אודות כל מטוס בצי המטוסים.

- כאשר נבחר את אחד ממאפייני הטיס-יוצג בצורת Line graph את ערכו של המשתנה לאורך הזמן מהרגע שבחרנו ב

- במקביל נציג Line graph נוסף שיציג את המאפיין הקורלטיבי ביותר אליו במידה וקיים כזה.

- גלאי חריגות:

- ביטוי ויזואלי לגלאי חריגות עבור קורלציה גבוהה.

- ביטוי ויזואלי לגלאי חריגות מבוסס מעגל עבור קורלציה בינונית.

- כאשר מתגלה חריגה אנו נציג אותה בהודעה.

רכיב הג'ויסטיק: רכיב זה מציג את התנועות שנעשות במטוס, הוא לא שולט עליהן בטאב זה.

slider :

- אנכי שמציג את מצב המצערת

- אופקי שמציג את מצב מייצב הכיוון

לוח שעונים: יציג מס' שעונים כגון גובה, מהירות, כיוון, אופק, פנייה, מהירות אנכית.

**Teleoperation**

יאפשר שליטה מרוחקת במטוס:

- בטאב זה יופיע הבקר ולוח השעונים

- המשתמש יוכל להזיז את הבקר ולהשפיע מרחוק על תנועת המטוס בצורה ידנית.

- המשתמש יוכל להזין קוד בשפת תכנות על מנת לגרום למטוס לנוע בצורה אוטומטית.

**Time Capsule**

יאפשר לשחזר ולתחקר טיסה שכבר הסתיימה – נקבל איזשהו קובץ csv מה-Backend לגביי טיסה שהמשתמש בוחר:

- המשתמש יוכל לבחור מתוך מאגר הטיסות שהתקיימו בעבר טיסה כלשהי ולהריץ אותה.

- יהיה פאנל של video controller שהמשתמש יוכל להזיז את הסרטון של הטיסה קדימה ואחורה, לעצור,לשלוט בזמן המהירות של הסרטון וכו.

- במקביל להרצת הטיסה יהיה לוח Monitoring שיראה מידע עדכני בהתאם לזמן שהמשתמש נמצא בו.

- מפעילים את Flightgear כמקרן בלי יכולות סימולטיביות, מזרימים לו את כל הנתונים שיש לנו מהקובץ שקיבלנו מה- Backend ושולחים את המידע לסימולטור וכתוצאה מהשידור הרציף לסימולטור נוכל לראות בשידור ישיר את נתוני הטיסה.

**ארכיטקטורת ה Frontend**

Diagram

Description automatically generated

**Model**– המודל מתקשר עם ה-Backend שמתקשר עם ה-Agent שמתקשר עם המטוס. ברגע שאנחנו נמצאים בטאב ה-Time capsule אנחנו נתחבר כ-Client ל- Flightgear ונזרים אליו נתונים כמו גובה, מהירות, כיוון ועוד ועל פי נתונים אלו הוא יצטרך למקם את המטוס בהתאם ולעדכן כל X שניות. (פותח socket ומקבל איזשהו port ו-ip ומתחבר אליו, אך במקרה זה בשונה מה-Agent אנחנו מכבים את היכולת הסימולטיבית וכך בעצם הוא יוצג כמקרן בלבד)

**View**–החלק שהמשתמש רואה ויכול לבצע בו פעולות (אפליקציית הדסקטופ שהסברנו עליה)

**ViewModel** - הוא מקשר בין ה-View ל-Model , כאשר מקבלים פקודה מהמשתמש ב-View ה-ViewModel מקבל אותה ומעביר את הפקודה ל- Model , קיים Data bindingבין ה- View ל- .ViewModelכאשר משהו משתנה או קורה באחד מהם - הוא מעדכן את השני.