

תאריך:\_\_\_\_\_\_\_\_\_\_\_

לכבוד

יחידת הפרויקטים מה"ט

\*\* יש לקרוא את הנספח להצעת הפרוייקט בקובץ הצעת פרוייקט PDF

**הצעה לפרויקט גמר**

**א. פרטי הסטודנטים**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| תאריך סיום  הלימודים | טלפון נייד | כתובת | ת.ז. 9 ספרות | שם הסטודנט |
| 2023 | 0524892070 | האנדרטה 2\1  באר שבע | 316505387 | קובי ביטון |
| 2023 | 0527557909 | אלי סיני 50/2 נתיבות | 205612955 | איתי אזולאי |

שם המכללה המכללה הטכנולוגית באר שבע. סמל המכללה: 72204 מסלול ההכשרה: הנדסאים.

מגמת לימוד: תוכנה

מקום ביצוע הפרויקט: המכללה הטכנולוגית באר שבע **ב. פרטי המנחה האישי**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מקום עבודה/תפקיד | תואר | טלפון נייד | כתובת | שם המנחה \* |
| המכללה  הטכנולוגית באר שבע | תואר שני | 052-4738238 | אביתר הכהן 7,  באר שבע | מר חרסונסקי מיכאל |

\* עבור מנחה אישי חדש יש לצרף קורות חיים, ניסיון מקצועי ותעודות השכלה לאישור מה"ט.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ חתימת המנחה האישי חתימת הגורם המקצועי מטעם מה"ט

* + - * 1. שם הפרויקט : ממשק ניהול מוסך “Pimp My Ride”

* + - * 1. רקע:

2.1 תיאור ורקע כללי:

המערכת משמשת לניהול המוסך: פתיחת כרטיסי עבודה, ניהול עובדים וניהול מלאי, לו"ז לקוחות ואפשרות לקביעת תור.

2.2 מטרות המערכת:

אפשרות לקביעת תור מצד לקוח

קבלת הצעת מחיר לגבי טיפול/ תיקון תקלה

פתיחת כרטיס עבודה

ניהול מלאי

ניהול עובדים )שעות,שכר וכו(

הזמנת חלפים/מוצרים מספקים

* + - * 1. סקירת מצב קיים בשוק, אילו בעיות קיימות:

כיום המצב במוסכים הוא כזה שלא מאפשר ללקוח לקבוע תור מבלי ליצור קשר טלפונית או להגיע פיזית למוסך, דבר שמכביד ומאריך את תהליך קבלת השירות מהמוסך. הצעת מחיר דיגיטלית במקום טפסי נייר.

* + - * 1. מה הפרויקט אמור לחדש או לשפר:

הפרויקט אמור להקל על השתמש לקבל שירות דיגיטלי מהמוסך, החל מקביעת תור דיגיטלי דרך

משך זמן טיפול ועד להצעת מחיר דיגיטלית, וכך הלקוח יכול לקבל החלטות מבלי לבזבז זמן מיותר במענה טלפוני או בהגעה למקום.

* + - * 1. דרישות מערכת ופונקציונאליות:

5.1 - דרישות מערכת:

המערכת תפעל ע"י תוכנת מחשב שמותאמת למערכת הפעלה WINDOWS שתמומש ע"י visual forms C#, בנוסף יהיה אפליקציית צד לקוח שתמומש ע"י REACT NATIVE. צד השרת ימומש ע"י C# , הנתונים יישמרו בבסיס נתונים מקומי) SQL SERVER( .

5.2 - דרישות פונקיונאליות:

רשימת דרישות המשתמש מהמערכת, מהן הפעולות בהן נדרשת המערכת לתמוך.

5.2.1 - דרישות משתמש )מכונאי( :

5.2.1.1 - דיווח שעות עבודה

5.2.2 - דרישות משתמש )יועץ שירות( :

5.2.2.1 - התחברות למערכת

5.2.2.2 - עדכון פרטיים אישיים וסיסמא

5.2.2.3 - דיווח שעות עבודה

5.2.2.4 - פתיחה וסגירה של הצעות מחיר וכרטיסי עבודה.

5.2.3 – דרישות משתמש )מנהל( :

5.2.3.1 – כל דרישות משתמש יועץ שירות ובנוסף:

5.2.3.2 – עדכון פרטיים אישיים לעובדים.

5.2.3.3 – הזמנת מלאים מספקים וניהול מלאי.

5.2.4 – דרישות מנהל מערכת

5.2.4.1 – הוספה\הסרה\עדכון של משתמשים

5.2.4.2 עריכת בסיס נתונים )יצרנים \ דגמים \ מלאי \ ספקים \ משתמשים(

6. בעיות צפויות במהלך הפיתוח ופתרונות)תפעוליות , טכנולוגיות , עומס ועוד:(

6.1 – תיאור הבעיות הללו כפועל יוצא של דרישות המשתמש מהתוכנה:

בעיה 1: כפילויות של הצעות מחיר לאותו הרכב.

6.1 – פתרון אפשרי לבעיה מס 1:

שמירת הצעות מחיר לפי מספר ייחודי של ההצעה ולא לפי מספר רכב.

7. פתרון טכנולוגי נבחר:

7.1 טופולוגית הפתרון- כלומר: פרישת המערכת , היכן יתבצע יישום המערכת) deployment(, מרכיבי הפרישה. הנ"ל ברמת מערכת )לדוג' פרויקט פיתוח אתר אינטרנט: המערכת מורכבת משרת, ממשק משתמש בצד הלקוח,, DB’s, טווח תקשורתי-אינטרנט, המערכת תיושם ברשת האינטרנט , יש להציג את דיאגרמת המערכת וכו ('

7.2 טכנולוגיות בשימוש.)איזה ומדוע בכמה מילים( C# - פיתוח באמצעות C# מאפשר תכנות ויזואלי לתוכנה שהולכת לרוץ על מערכת ההפעלה של ווינדוס.

–Sql server בסיס נתונים שיתחבר לC#.

7.3 שפות הפיתוח:)איזה שפות ומדוע בכמה מילים(?

#C 7.3.1

7.4 תיאור הארכיטקטורה הנבחר ת- הסבר בכמה מילים מדוע הארכיטקטורה הנבחרת היא חלוקה ל 3 שכבות – 3 Tier Architecture

הפרויקט מבוסס על database ולכן מומלץ להשתמש בגישת מודל השכבות.

לגישה יש יתרון של תחזוקה קלה. בשל החלוקה לשכבות ניתן לבצע שינויים בשכבה כלשהי בקלות ומבלי להשפיע על השכבות האחרות.

7.5 חלוקה לתכניות ומודולי ם

שכבת התצוגה: שכבת ממשק המשתמש

שכבת הלוגיקה העסקית: שכבת עיבוד המידע ושליחתו לשכבת התצוגה שכבת הנתונים: שכבה של מסד נתונים האחראית על שמירה עדכון והוספה של מידע.

7.6 סביבת השרת )מקומי, וירטואלי, ענן, שירות אירוח( שרת מקומי ותחנות עבודה המחוברות באותה הרשת

7.7 ממשק המשתמש/לקוח – GUI מוצג על גבי תוכנה של ווינדוס.

7.8 ממשקים למערכות אחרות/ :API הדפסת הצעות מחיר + כרטיסי עבודה + תלושי שכר + דוחות + חשבונית.

7.9 שימוש בחבילות תוכנה:

#C

8.0 שימוש במבני נתונים וארגון קבצים

8.1 פירוט מבני הנתונים

**Employee**: Id, FullName, ContactInformation, JobDefinition

**EmployeeEntry**: EmployeeId, EntryTime, ExitTime

**WorkTicket**: CarId, Complaint, WorkTicketOfferId, EmployeeId

**WorkTicketOfferId**: WorkTicketOfferId, CarId, PartsLayout, WorkLayout

**PartsCollection**: PartId, PartName, CostToConsumer, CostToGarage,Quantity, SafeQuantity

**ClientsCollection**: CarId, CarOwnerName,CarOwnerContacts, CarHistoryTable

**CarHistoryTable**:CarId, WorkTicketHistory,Warranty

**SupplliersCollection**: SupplierId, SupplierName, SupplierContact

**SuppliersOrders**: OrderId, OrderInformation, OrderDate, SupplierId

**ScheduleCollection**: CarId, Date,WorkTicketOfferId

8.2 נא פרט את שיטת האיחסון )מאגר, קבצים ובאיזה טכנולוגיה( בסיס נתונים SQL Server

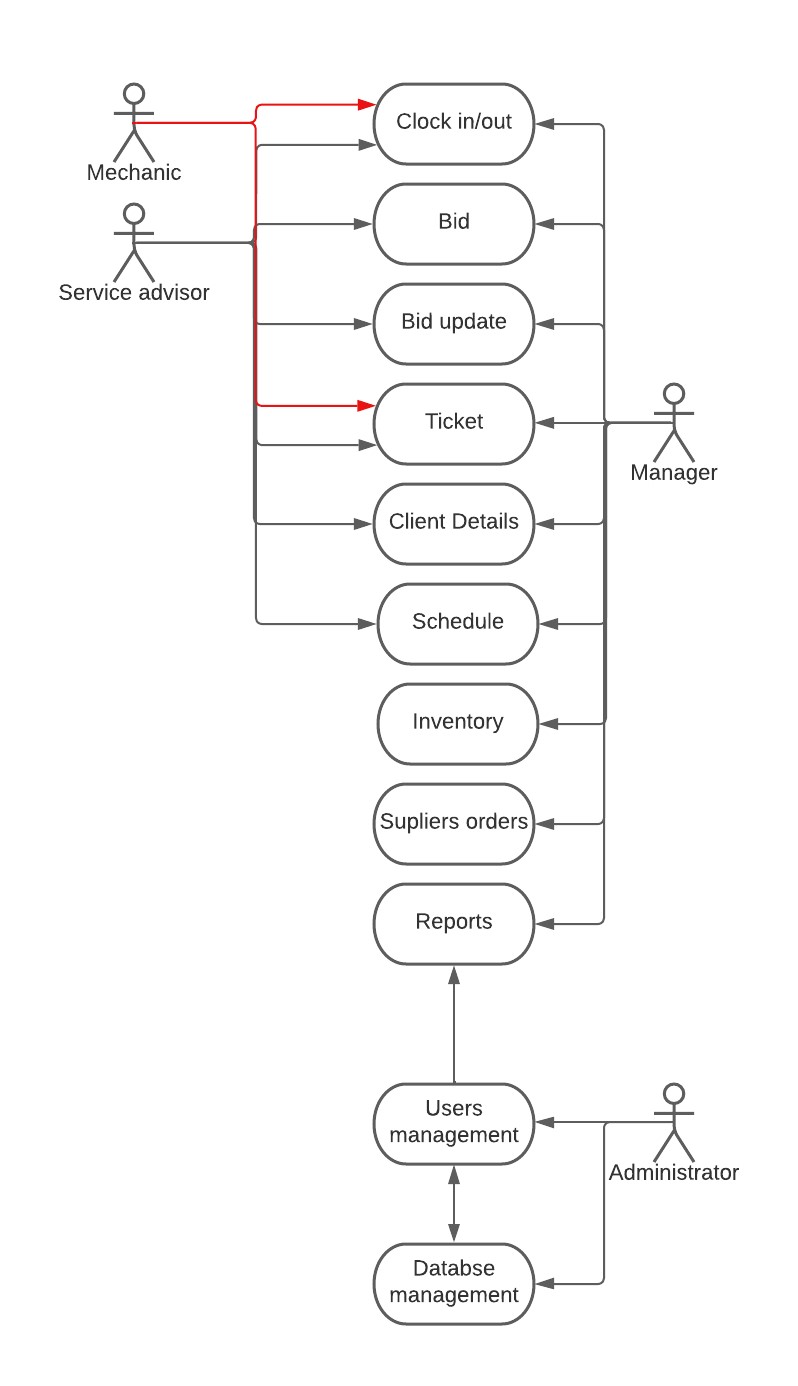
8.3 נא ציין מנגנוני התאוששות מנפילה/קריסה/תמיכה בטראנזקציות.

המערכת מיועדת לשימוש בארגונים קטנים ומספר הפניות במקביל למסד הנתונים

והשרת אינו צפוי להיות גדול כלל.

9

Use Case 9.1



1. תיאור המרכיב האלגוריתמי - חישובי
   1. איזה בעיה בא לפתור, איך יפתור?

אלגוריתם ניהול מלאי יסייע במניעת כפל הזמנות בנוסף יתריע על מלאי שעומד להיגמר.

אלגוריתם ניהול כרטיסי עבודה שתפקידו למנוע כפילויות של כרטיסי עבודה.

* 1. איסוף מידע וניתוחים סטטיסטיים )אנליטיות(

חישוב שעות עבודה לעובד.

חלקים שנצרכו ועלויות ממוצעות לתקופת זמן.

ניתוח הזמנות מספקים כולל דוחות של עלויות פר תקופת זמן.

דוחות הכנסות והוצאות של המוסך.

1. תיאור / התייחסות לנושאי אבטחת מידע

פרטי משתמשים רגישים וסיסמאות יוצפנו באמצעות ספריית security.Cryptography.

המערכת מותקנת על שרת מקומי וללא גישה לרשת.

1. משאבים הנדרשים לפרויקט:
   1. מספר שעות המוקדש לפרויקט, חלוקת עבודה בין חברי הצוות:

סה”כ 350 שעות עבודה.

חלוקי שווה בין חברי הצוות.

* 1. ציוד נדרש:

מחשבים לכתיבת קוד.

* 1. תוכנות נדרשות:

Visual studio code

SQLServer

* 1. ידע חדש שנדרש ללמוד לצורך ביצוע הפרוייקט:

התממשקות להדפסה מתוך התוכנה.

* 1. ספרות ומקורות מידע:

אינטרנט.

מרצה.

1. תוכנית עבודה לשלבים למימוש הפרויקט:

יזום הרעיון.

ניתוח מערכת.

ניתוח מבנה נתונים.

כתיבת קוד מערכת.

עיצוב.

בדיקות תוכנה.

1. תכנון הבדיקות שיבוצעו:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| חשיבות | ידנית / אוטומטית | מקרה הבדיקה | מספר הדרישה במס מך האפיון | מספר |
|  | ידנית | דיווח שעות עבודה | 5.2.1.1 | 1 |
|  | ידנית | עדכון פרטים אישיים | 5.2.2 | 2 |
|  | ידנית | פתיחה וסגירה של הצעות מחיר וכ רטיסי עבודה )וידוא שדות, וידוא פונ קציונליות תקינה, בדיקת כפל, בדיקת חישוב עלויות תיקון | 5.2.2.4 | 3 |
|  | ידנית | הזמנת מלאים / התראה על מלאי שעומד להיגמר | 5.2.3.3 | 4 |
|  | ידנית | מנהל מערכת- ניהול בסיס נתונים וניהול משתמשים | 5.2.4 | 5 |

1. )version control (בקרת גרסאות

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ חתימת המנחה האישי

1. *הערות ראש המגמה במכללה*

\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_מאושר\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **אישור ראש המגמה**



**שם: \_\_\_\_\_\_ברגמן איגור\_\_\_\_\_\_\_ חתימה** **: \_\_\_\_\_\_\_\_\_\_\_\_ תאריך:\_\_\_\_ 23/10/23\_\_\_**

1. **הערות הגורם המקצועי מטעם מה"ט**

\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **אישור הגורם המקצועי מטעם מה"ט**



**שם: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ חתימה: \_\_\_\_\_\_\_\_\_\_\_\_ תאריך:** 30/01/2023**\_\_\_\_\_\_\_\_\_\_\_\_\_**



סימוכין : כלל 2

**הצהרת סטודנט**

סטודנט 1 – שם : איתי אזולאי תעודת זהות : 205612955

סטודנט 2 – שם : קובי ביטון תעודת זהות : 316505387

שם המכלל בה לומדים הסטודנטים: **המכללה הטכנולוגית באר – שבע קדימה מדע (ע"ר)**

אנו החתומים מטה מצהירים בזאת כי פרויקט הגמר וספר הפרויקט המצ"ב נעשו על ידינו בלבד.

פרויקט הגמר נעשה על סמך הנושאים שלמדנו במכללה ובאופן עצמאי.

פרויקט הגמר וספר הפרויקט נעשו על בסיס הנחייתו של המנחה האישי.

מקורות המידע בהם השתמשנו לביצוע פרויקט הגמר מצוינים ברשימת המקורות בספר הפרויקט.

אנו מודעים לאחריות שהננו מקבלים על עצמנו על ידי חתימתנו על הצהרה זו שכל הנאמר בה את ורק אמת.

סטודנט 1 – חתימה: תמונה שמכילה שרטוט, ציור, אנטנה, עיצוב

התיאור נוצר באופן אוטומטי תאריך: 23/10/2023

סטודנט 2 – חתימה: תמונה שמכילה שרטוט, קולב

התיאור נוצר באופן אוטומטי תאריך: 23/10/2023

**אישור המנחה האישי**

הריני מאשר שהפרויקט בוצע בהנחייתי, קראתי את ספר הפרויקט ומצאתי כי הוא מוכן לצורך הגשת הסטודנטים להגנה על פרויקט הגמר.

תמונה שמכילה טקסט, גופן, צילום מסך, לבן

התיאור נוצר באופן אוטומטי

**תקציר הפרויקט Pimp my ride**

הפרויקט הוא מערכת ניהול מוסך רכבים הפועלת בסביבת Windows ומשתמשת בשפת תכנות C# עם ממשק משתמש באמצעות WinForms. המערכת נוצרה כדי לספק פתרון נוח ויעיל לניהול ותחזוקת רכבים במוסך.

התוכנה מטרתה לספק ייעול בניהול ובתחזוקת הרכבים, כולל ניהול לקוחות, רכבים, פעולות תחזוקה, ותיקונים.

הפרויקט כולל בתוכו:

1. **ניהול לקוחות**:

- הוספה, מחיקה ועדכון של פרטי לקוחות.

- צפייה ברשימת לקוחות ופרטיהם.

2**. ניהול רכבים**:

- הוספה ועדכון של פרטי רכבים, כולל דגם, שנת ייצור, ומספר רישוי.

- תצוגה וניהול של רשימת רכבים במוסך.

3. **תחזוקה ותיקונים**:

- רישום פעולות תחזוקה ותיקונים לכל רכב.

- היסטורית טיפולים עבור רכבים שטופלו במוסך.

4. **ממשק משתמש ידידותי**:

- עיצוב והפעלה נוחים של ממשק משתמש באמצעות WinForms.

- קלות בשימוש, כולל תפריטים וכפתורים לפעולות נפוצות.

5. **אחסון נתונים**:

- שימוש בממסד נתונים SQL Server שרץ על שרת מקומי.

**טכנולוגיות שוטפות**:

הפרויקט משתמש בשפת תכנות C# עם סביבת פיתוח Visual Studio ומשתמש ב WinForms כטכנולוגיה לבניית ממשק המשתמש.

**השגחה על קוד**:

הקוד נכתב בסגנון נקי וארגוני, כולל תיעוד רלוונטי להבטחת קריאות ותחזוקה קלה בעתיד.

**לסיכום**:

המערכת מספקת פתרון מתקדם ויעיל לניהול מוסך רכבים, מסייעת בשמירה על רשומות נכונות ומספקת ממשק משתמש ידידותי המאפשר ניהול יעיל ונוח של פרטים ופעולות שונות.

**תוכן עניינים:**

1. מבוא.
2. עקרונות התכנון.
   1. בניית Controllers ו – handlers לכל ישות (טבלה) בבסיס הנתונים.
   2. שימושים בספריות.
   3. בניית מסכים בממשק המשתמש.
   4. מסכים המותאמים לפי הרשאות המשתמש.
3. תיאור העבודה
   1. ממשק המשתמש.
   2. צד השרת.
   3. בסיס הנתונים.
4. מפת מודלים.
5. מסכי המשתמש
   1. Login
   2. Dashboard
   3. Tickets
      1. יצירת כרטיס עבודה חדש.
      2. כל הכרטיסים הפתוחים.
      3. יצירת הצעת מחיר.
      4. הצעות מחיר פתוחות.
   4. Clients
      1. חיפוש לקוח.
      2. טבלת כל הלקוחות.
      3. יצירת לקוח חדש.
   5. Storage
      1. טבלת כל הספקים.
      2. יצירת ספק חדש.
      3. יצירת הזמנה מספק.
      4. כל החלקים במחסן.
      5. כל ההזמנות שבוצעו.
   6. Admin
      1. יצירת משתמשים חדשים.
      2. עמוד תחזוקה הכולל בתוכו הכנסת מידע למערכת.
      3. הכנסת דגמי רכבים חדשים למערכת.
6. קוד המערכת
   1. ממשק המשתמש.
   2. שרת העיבוד.
   3. בסיס הנתונים.
7. **מבוא**:

בעידן המודרני, המערכות הממוחשבות תובנה נוספת ומעמיקה בכמעט כל תחום של חיינו. עם עליית הטכנולוגיה, הצורך בכלים דיגיטליים לניהול יעיל וממוקד של מגוון תחומים עולה. אחד מתחומים אלו הוא תחום ניהול הרכבים והמוסכים, שבו מתעסקים בשולי הטכנולוגיה וההנדסה יחד.

בהקפדה על האספקטים השונים של ניהול ותחזוקת הרכב, החלטנו לפתח מערכת ניהול מוסך רכבים באמצעות תוכנה. המערכת שפותחה בשפת תכנות C# באמצעות סביבת הפיתוח Visual Studio ובעזרת טכנולוגיית WinForms, מיועדת להקל ולשדרג את תהליכי הניהול והתחזוקה של רכבים במוסך.

**מטרה**:

המטרה העיקרית של המערכת היא ליצור כלי יעיל וידידותי למשתמש המסייע בניהול ובמעקב אחר פעולות התחזוקה והתיקונים של רכבים במוסך. באמצעות ממשק משתמש אינטואיטיבי ותכונות יעילות, המערכת מאפשרת למנהלי המוסך ולצוות התחזוקה לשמור על מעקב מדויק ומעודכן של רכביהם ולשרת את לקוחותיהם בצורה מקצועית ויעילה.

**תכנון ומבנה**:

התוכנה פותחה בהתבסס על ארכיטקטורת קוד יציבה וקלה לתחזוקה, ותומכת בניהול רשומות מקומי באמצעות בסיס נתונים. בניית הממשק נעשתה באמצעות WinForms, מה שמבטיח יעילות ונוחות בשימוש.

בעזרת הפרויקט, אנחנו מקווים לתרום למתקדמים בתחום ניהול הרכב והמוסך ולהפוך את תחום זה ליותר יעיל וזמין, עם מערכות מתקדמות וידידותיות למשתמש.

1. **עקרונות התכנון:**

בחרנו להשתמש בארכיטקטורת REST API אשר משתמשת במודל שפשוט, יעיל ונפוץ עבור שירותים רשתיים. הנה סיכום קצר על טעמי לבחירה זו:

**יעילות ניהולית:**

- REST (Representational State Transfer) משתמשת במודל יעיל וקל שהקלט שלו קריא וברור. זה מפשט את התקשורת בין השרת והלקוח, וגורם לפרויקט להיות נגיש וקל לתחזוקה.

**סטנדרטיזציה**:

- REST הוא סטנדרט נפוץ כאשר הוא חלק מהאינטרנט ומערכת ה-WWW. זה מאפשר איכות קוד גבוהה, נגישות ושיבוץ נתונים בצורה יחידה ואחידה בפרויקט.

**תקשורת גמישה**:

- REST מספק גמישות בתקשורת, וזהו יתרון חשוב בפרויקטים שבהם ישנם מספר פלטפורמות ולקוחות שונים.

**אוטונומיה בכתיבה**:

- באמצעות REST, כל רשות בקצה הלקוח והשרת יכולה לפעול באופן עצמאי ולא תלויה יותר מדי בשינויים בצד השני. זה מקנה ניתוח והתקנות באופן נפרד.

**תמיכה בפורמטים פתוחים**:

- REST מתמקדת בשימור קריאות ונגישות של המידע, בדרך כלל תחת פורמטים פתוחים כמו JSON או XML. זה מאפשר ללקוחות ולשרתים שונים להבין ולתקשר בצורה יעילה.

בחירת REST API מציינת את המחויבות לבנות מערכת יעילה, קלה לתחזוקה, ומותאמת לכמה שיווקים ופלטפורמות ככל שזה אפשרי. היא מספקת פתרון טכנולוגי פשוט ויעיל שמתמקד בהעברת מידע בצורה ברורה וקריאה, עומדת בסטנדרטים נפוצים, ומאפשרת תקשורת נגישה ויעילה בין רכיבי המערכת.

2.1 **Controllers and Handlers**

**Controllers (בקרים)**:

- המחלקות האחראיות על לקבל ולתת קלט ופלט למערכת.

- כל בקר מתמקד בתפקיד מסוים, כגון ניהול לקוחות, רכבים, או פעולות תחזוקה.

- מטרתם העיקרית היא להפנות בקשות הגעה מהלקוחות לפונקציות הנכונות לטפל בהן.

2. **HANDLERS (טפלים:**

- מחלקות אשר מיועדות לטפל בבקשות קצה, כל טפל יורש ממחלקה אבסטרקטית אשר מיועדת לטפל במקרה ספציפי.

- נוצרו כך שכל הטפלים ייכתבו לפי נהלים מוגדרים מראש, מה שמבטיח יחידות קוד, קריאות, ותחזוקה יעילה.

המבנה שלנו נבנה בצורה אשר בה הוא נוקט בגישה פונקציונלית, קצרה וברורה, המקנה את היכולת לפתח בקלות פונקציות חדשות ולתחזק את הקוד בצורה יעילה. המודל שנבחר מסייע לנו לקיים כתיבת קוד בצורה אחידה ומאורגנת היטב בפרויקט, וכמובן מוודא שכל הקוד עומד בנהלים ובסטנדרטים שהוגדרו מראש.