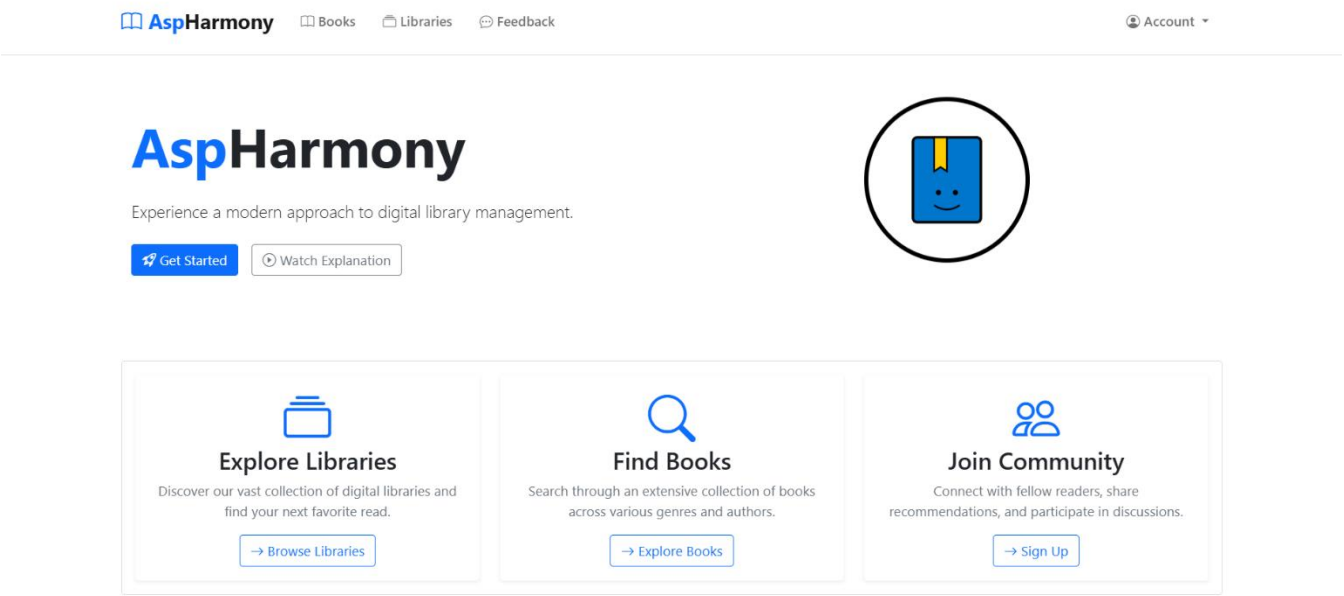
עבודת גמר בתכנון ותכנות מערכות
במסלול שירותי רשת אינטרנט (תשפ"ה)

ניהול ספרים וספריות



שם התלמיד: לירון כנר

ת"ז: XXXXXXXXX

בית ספר: כי"ח אליאנס – חיפה

שם המורה: ולרי קריפוקס

אליאנס
אדם | חברה | סביבה

תוכן עניינים

מבוא

מטרות המערכת

1. **לספק ניהול ספרים דיגיטליים** – משתמשים מסוגלים ליצור ולצפות בספרי הפרויקט. יתר על כן, משתמש אשר יצר ספר נחשב כיוצר הספר; בנוסף ליכולות משתמש יוצר מסוגל לערוך ולמחוק את הספר.

2. **לספק ניהול ספרייה דיגיטלית** – משתמשים מסוגלים ליצור, להצטרף, לצאת, להשאיל ולהחזיר ספרים בספרייה. יתר על כן, משתמש אשר יצר ספרייה נחשב כמנהל הספרייה; בנוסף ליכולות משתמש מנהל מסוגל לערוך, למחוק, להוסיף ספרים ולנהל את חברי הספרייה, כלומר למחוק ולקדם אותם.

3. **לספק גמישות אחסון מידע של ספרים** – יוצר ספר מסוגל לאחסן את מידע הספר שיצר במספר "שרתים" (ראו עמוד 12). מערכת זו נועדה על מנת לאפשר למשתמש לבחור שיטת אחסון המתאימה לצרכיו.

קהל היעד

קהל היעד מורכב מחובבי ספרים וקוראים נלהבים, ספריות ומוסדות חינוך, מועדוני ספרים וקבוצות קריאה ואנשים המעוניינים לארגן ולשתף את אוסף הספרים האישיים שלהם.

תיאור האתר

האתר מספק פלטפורמה דיגיטלית מתקדמת לניהול ואחסון ספרים וספריות. המשתמשים יכולים לצפות באוספי הספרים שבאתר ולבצע חיפושים מתקדמים כדי למצוא ספרים שמתאימים לתחומי העניין שלהם. האתר מאפשר יצירת ספרים וספריות, מה שמאפשר בנייה של קהילות קריאה ותמיכה בניהול ספרים משותף. בנוסף, האתר מציע ממשק גמיש לאחסון ואבטחת מידע על ספרים, המותאם לצרכים השונים של המשתמשים, כמו גם מערכת ניהול מתקדמת להוספה, עריכה, ומחיקת ספרים ומעקב אחר פעילויות השאלה והחזרה. האתר שם דגש על חווית משתמש אישית, עם אפשרויות התאמה שונות לקבלת עדכונים והודעות, וכלים מתקדמים לניהול ספריות דיגיטליות באופן ידידותי ומותאם אישית.

תפקידים באתר

אורח: תפקיד עבור לקוחות שלא ביצעו תהליך התחברות לאתר. האורח יכול:

- להתחבר
- להירשם

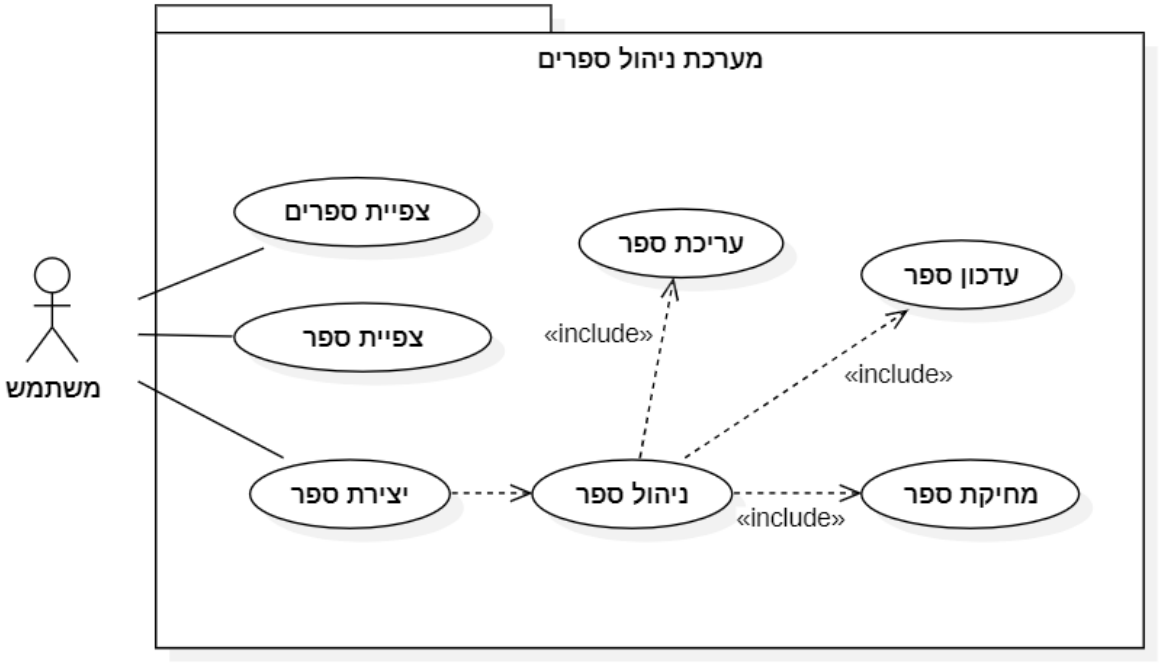
משתמש: תפקיד עבור לקוחות שביצעו תהליך התחברות לאתר. משתמש יכול:

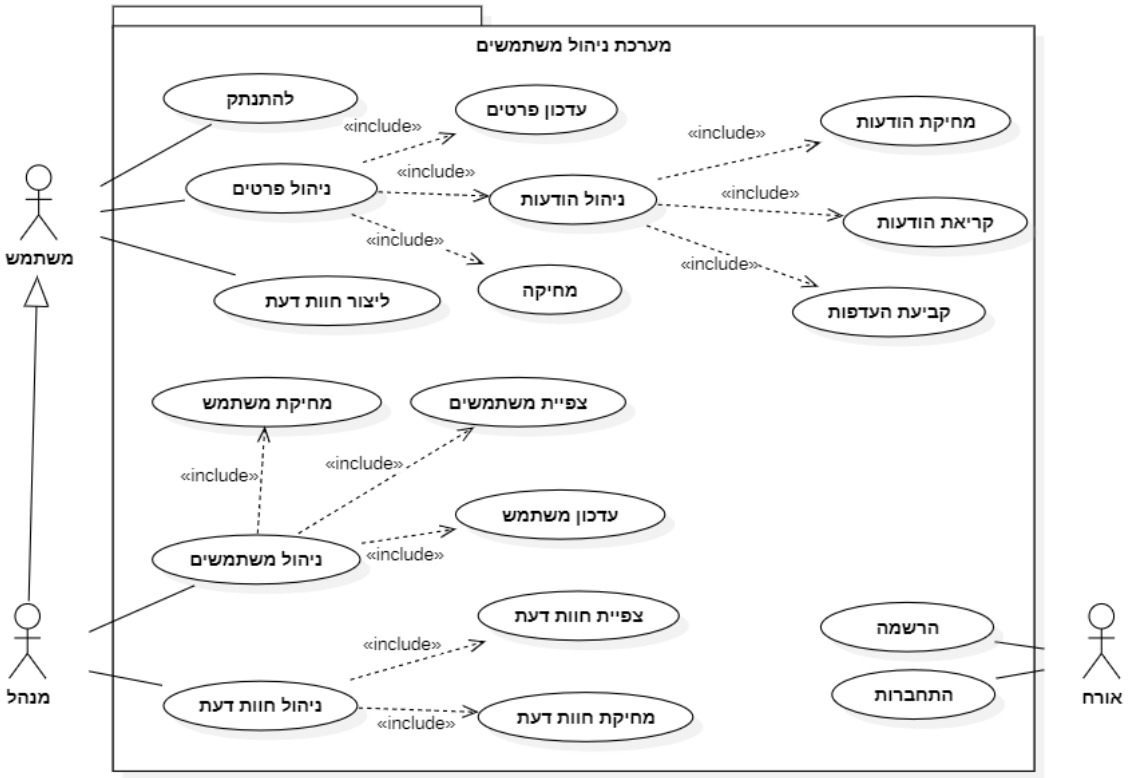
- לנהל פרטים אישיים
- להתנתק
- לתת חוות דעת על האתר
- לצפות וליצור ספרים ובנוסף למחוק ולערוך ספרים שיצר
- לצפות וליצור ספריות ובנוסף למחוק ולערוך ספריות שיצר
- לשאול ולהחזיר ספרי ספרייה
- לצפות, למחוק ולקבוע כיצד לקבל את הודעות האתר

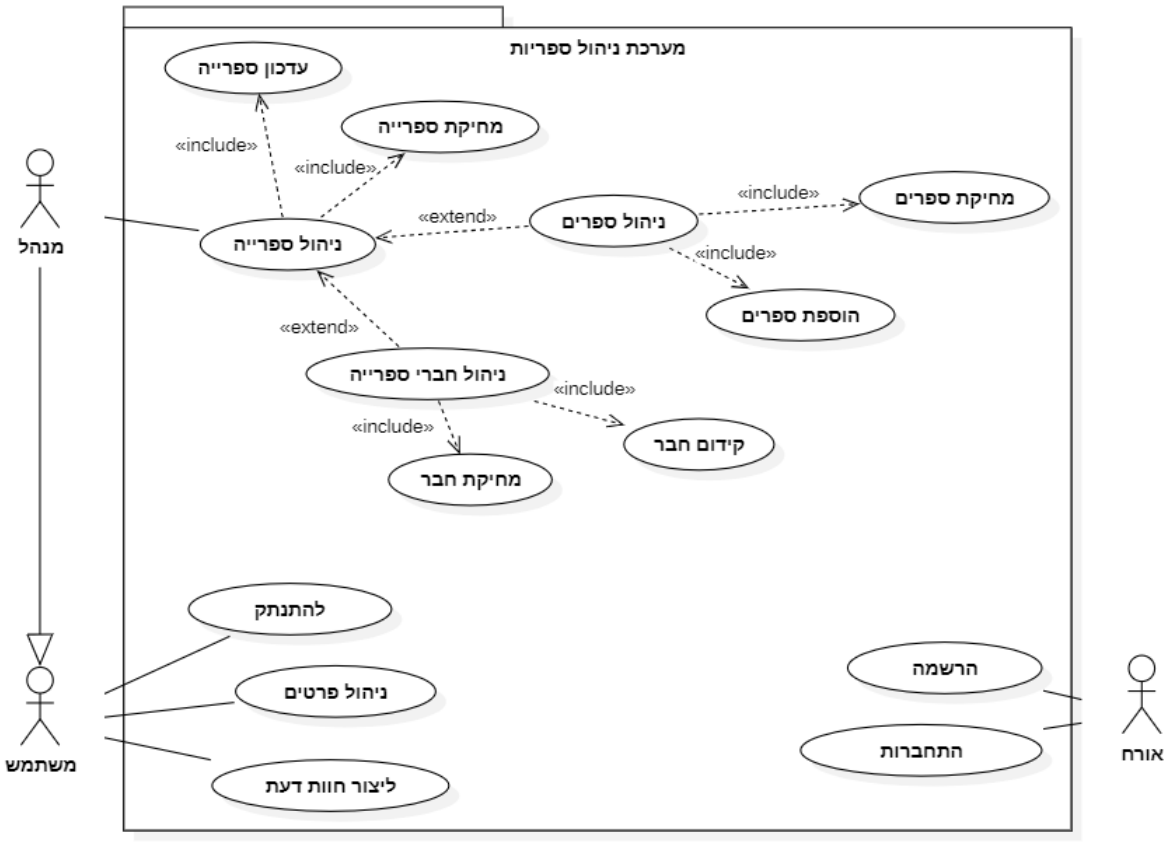
מנהל: תפקיד עבור משתמשים בעלי הרשאת גישה גבוהה. מנהל יכול:

- לצפות, לעדכן, למחוק ולערוך משתמשים
- לצפות ולמחוק חוות דעת על האתר

דיאגרמות Use Case






מערכת ניהול ספריות
עדכון ספרייה
מחיקת ספרייה
«include»
«include»
מנהל
ניהול ספרייה
«extend»
ניהול ספרים
«include»
מחיקת ספרים
«include»
הוספת ספרים
«extend»
ניהול חברי ספרייה
«include»
«include»
קידום חבר
מחיקת חבר
להתנתק
ניהול פרטים
ליצור חוות דעת
משתמש
הרשמה
התחברות
אורח

מבנה בסיס הנתונים

בפרויקט השתמשתי בספרייה הנקראת Entity Framework כך הטבלאות במסד הנתונים נוצרות מן מחלקות בקוד.

המחלקה ApplicationContext יורשת מן המחלקה IdentityDbContext (מחלקה מן ASP.NET Identity) ומציינת את הקשר למבנה הנתונים ובעזרתה ניתן לבצע פעולות על המבנה נתונים.

בנוסף כתבתי את המחלקה AdoContext, מחלקה עבור ביצוע פעולות על מסד הנתונים.

להלן הסבר על הטבלאות:

Libraries - טבלת הספריות.

Books – טבלת הספרים.

LibraryBooks – טבלת קשר המחברת בין ספר לבין ספרייה.

BooksLoans – טבלה קשר ההשאלות הספרים שבספרייה.

LibraryMembership – טבלת קשר בין משתמש לבין ספרייה.

AspNetUsers – טבלת משתמשים סטנדרטים מן ASP.NET Identity.

AspNetRoles – טבלת התפקידים שבאתר.

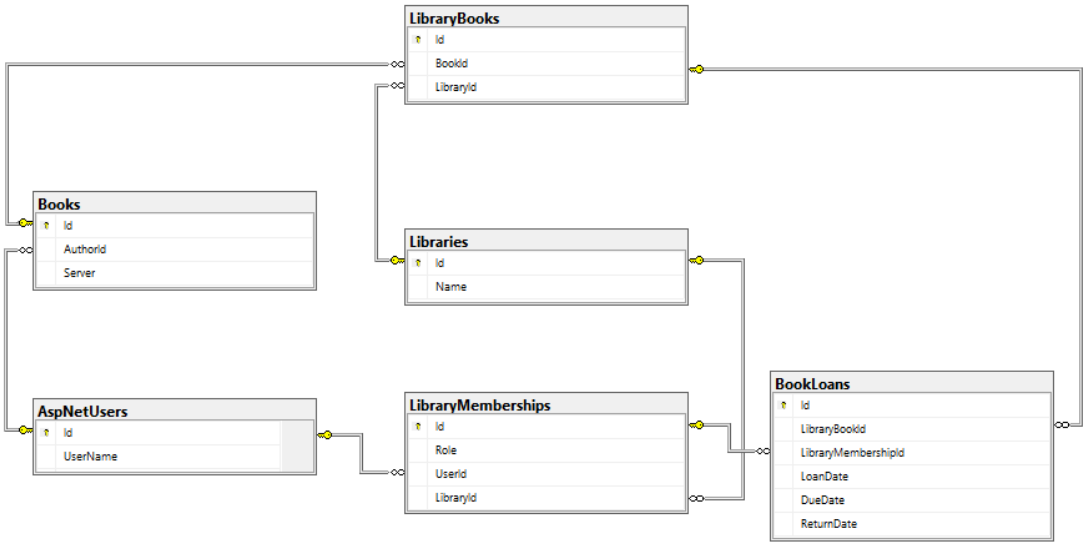
AspNetUserRoles – טבלת קשר בין משתמש לבין תפקיד באתר.

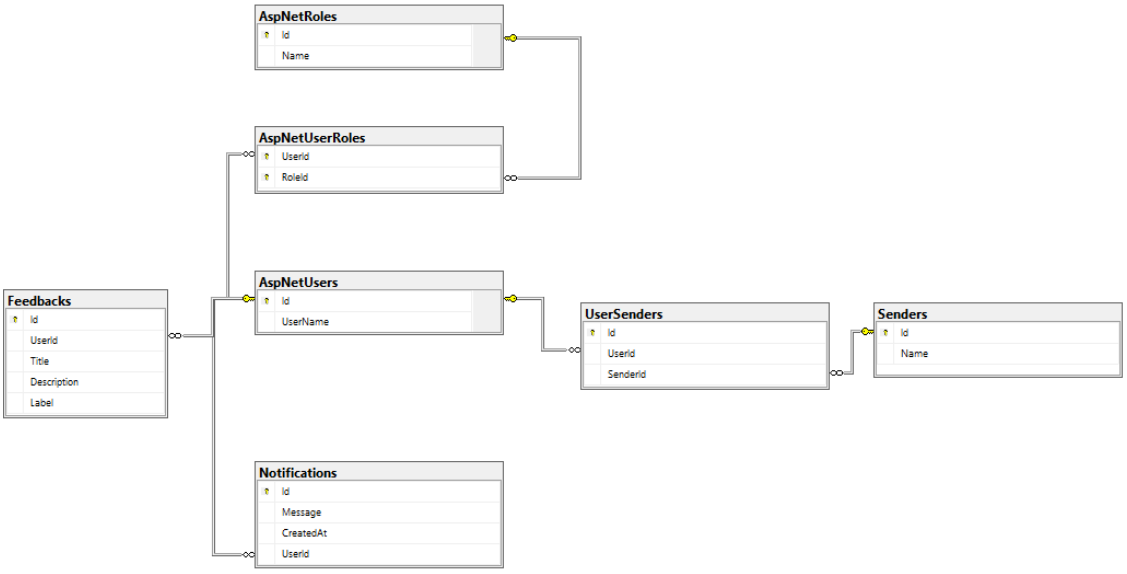
Notifications – טבלת התראות משתמש.

Senders – טבלת מתקשרים.

UserSenders – טבלת קשר בין מתקשר למשתמש.

התרשים הראשון מציג את הקשרים בין הטבלאות העיקריות שבמערכת. בעוד והתרשים השני מציג את מבנה הטבלאות של המשתמשים תפקידים וההתראות.





מבנה הפרויקט

Data Access Layer

שכבה זו אחראית על הגישה למסד הנתונים ומכילה מודלים (Entities) מחלקות המייצגות את טבלאות בסיס הנתונים, מאגרים (Repositories) מחלקות המבצעות פעולות על טבלאות בסיס הנתונים ובנוסף מחלקות הניגשות למסדי נתונים. שכבה זו משתמשת ב Entity Framework.

Business Logic Layer

שכבה זו אחראית על הלוגיקה העסקית של האתר. היא מטפלת בעיבוד נתונים, אכיפת כללים עסקיים וקואורדינציה בין שכבות שונות של האתר. השכבה מכילה שירותים (Services) מחלקות הלוגיקה, אירועים (Events) מחלקות המאזינות לאירועים השונים שבאתר ושרתים.

Presentation Layer

שכבה זו אחראית על תצוגת האתר במבנה. MVC הקונטרולרים (Controllers) מקבלים בקשות מהמשתמש, מעבדים אותן ומכינים את הנתונים להצגה. התצוגות (Views) אחראיות על הצגת המידע למשתמש בפורמט HTML ועל קבלת קלט מהמשתמש. מודלי התצוגה (View Models) משמשים להעברת נתונים בין הקונטרולרים לתצוגות ומגדירים את מבנה הנתונים הספציפי הנדרש לכל תצוגה.

Web Services

פרויקט זה נועד לשמש כממשק לשירותי רשת של המערכת. השימוש ב SOAP-מאפשר תקשורת מבוססת XML עם מערכות אחרות שתומכות בפרוטוקול זה.

Web Api

פרויקט זה מיועד לשמש כממשק API של המערכת. הוא מאפשר גישה לפונקציונליות של המערכת דרך נקודות קצה ,RESTful מה שמאפשר אינטגרציה עם מערכות חיצוניות או פיתוח של אפליקציות לקוח נפרדות.

Web Sockets

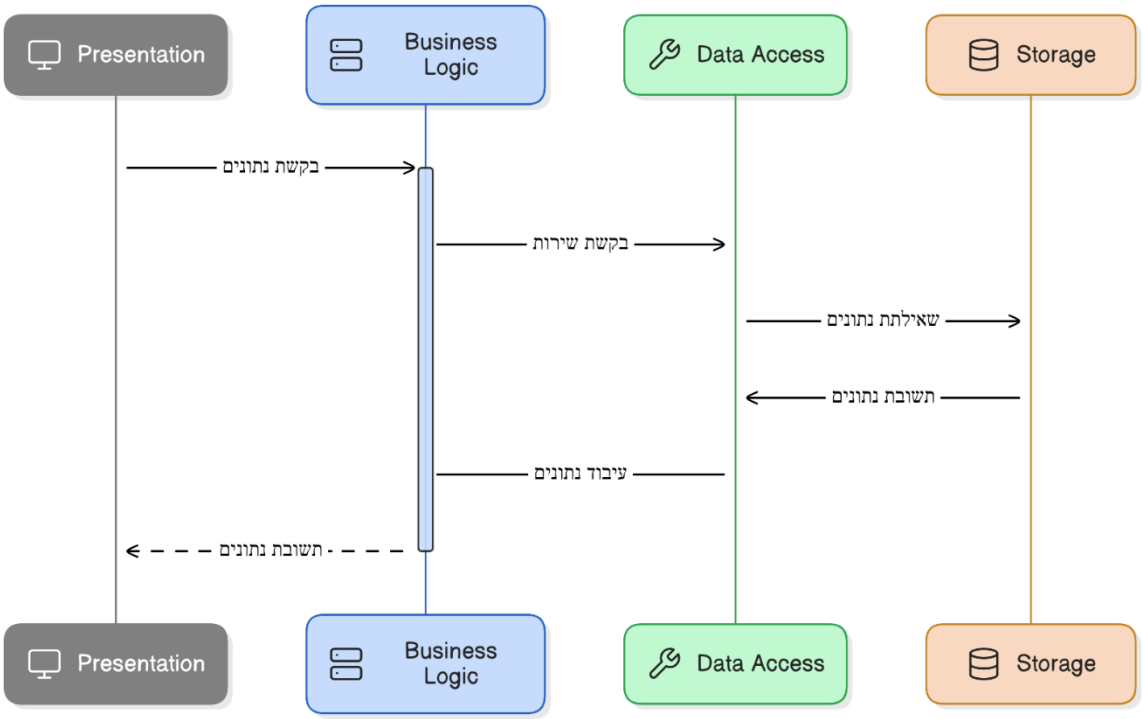
פרויקט זה מיועד לממש כממשק של Web Sockets של המערכת. השימוש בWeb Sockets מאפשר תקשורת בזמן אמיתי.

Utils

פרויקט זה מכיל כלים ופונקציות שימושיות שיכולות לשמש את כל חלקי המערכת. לדוגמה, שירותי הצפנה, חריגים, ופונקציות עזר אחרות.

Storage

פרויקט זה אחראי על אחסון ומכיל דפי מידע ומבני נתונים.



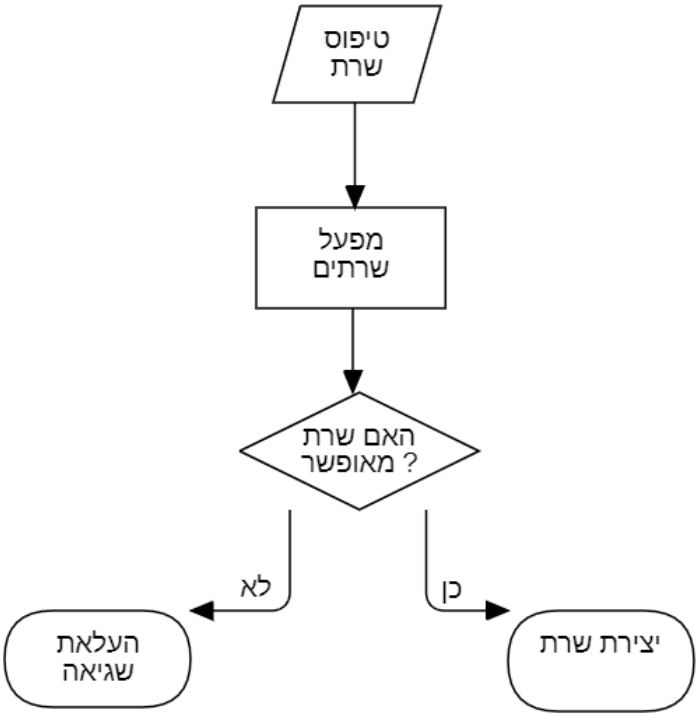
בקרת גרסאות

במהלך בניית הפרויקט השתמשתי בgit, קוד הפרויקט.

שרתים

שרת הוא דרך לאחסן את מידע הספר, בחרתי לא לאחסן את המידע במבנה הנתונים הראשי על מנת להדגים גמישות ארכיטקטונית וטיפול ייחודי בנתוני הספרים, בנוסף גישה זו מאפשרת למשתמשים לבחור בצורת האחסון המתאימה לצרכיהם.

ארכיטקטורה



טיפוס שרת

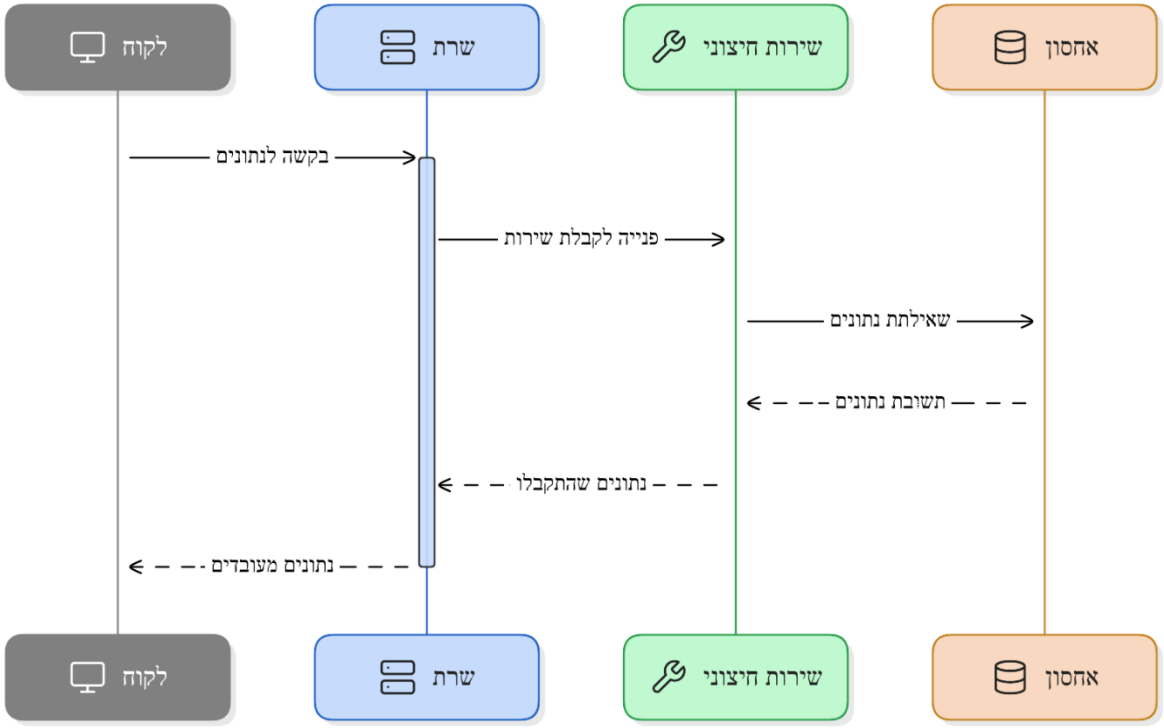
מציין את סוג השרת המבוקש. טיפוס השרת מייצג את הקלט למחלקת המפעל שמציינת איזה סוג של מופע שרת יש ליצור.

מפעל שרתים

מחלקת המפעל לוקחת את סוג השרת ומעבדת את הבקשה כדי לקבוע את השלבים הבאים ליצירת מופע שרת.

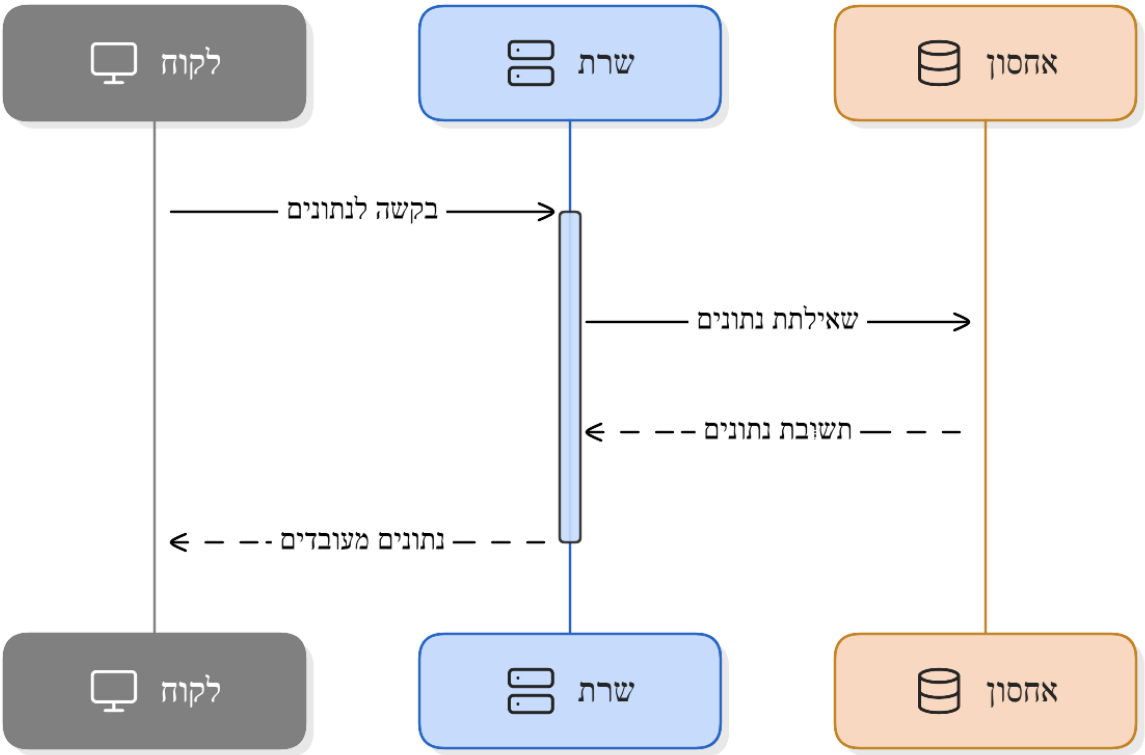
המפעל מבצע בדיקה האם השרת מאופשר לפי הקונפיגורציה של הפרויקט. במקרה שלא המחלקה מעלה שגיאה, אחרת המחלקה מחזירה מופע של השרת אשר התבקשה לייצר.

```
1  "DisabledServers": [ "Orion", "Atlas", "Nimbus2" ],
```



שרת בעל גישה דרך שירות חיצוני

1. **לקוח** - הלקוח שולח בקשה לנתונים מהשרת.
2. **שרת** - השרת מקבל את הבקשה מהלקוח ומבצע פנייה לשירות חיצוני לקבלת שירות.
3. **שירות חיצוני** - השירות החיצוני שולח שאילתת נתונים לאחסון.
4. **אחסון** - האחסון מחזיר את תשובת הנתונים לשירות החיצוני.
5. **שירות חיצוני** - השירות החיצוני מעביר את הנתונים שהתקבלו בחזרה לשרת.
6. **שרת** - השרת מעבד את הנתונים שקיבל מהשירות החיצוני ומחזיר אותם ללקוח.
7. **לקוח** – מציג את הנתונים.



שרת בעל גישה ישירה לאחסון

1. **לקוח** - הלקוח שולח בקשה לנתונים מהשרת.
2. **שרת** - השרת מקבל את הבקשה מהלקוח ומבצע שאילתת נתונים ישירות מול האחסון.
3. **אחסון** - האחסון מחזיר תשובת נתונים ישירות לשרת.
4. **שרת** - השרת מעבד את הנתונים ומחזיר ללקוח את התשובה הסופית.
5. **לקוח** – מציג את הנתונים.

סיכום

הדיאגרמות מתארות שני סוגים שונים של שרתים שבאתר:

- **שרתים בעלי גישה ישירה:** השרת מתקשר ישירות מול האחסון, שולח שאילתת נתונים ומקבל תשובות ישירות.
- **שרתים בעלי גישה עקיפה**: השרת נעזר בשירות חיצוני שמבצע את הבקשה מול האחסון ומחזיר לו את המידע.

ארכיטקטורה זו מאפשרת לנו גמישות ומגוון רחב של טכנולוגיות, בהמשך אסביר כי השרתים בעלי גישה עקיפה משתמשים בAPI ובשירותי רשת ובנוסף אראה שצורת האחסון יכולה להתבצע בעזרת מסד נתונים, xml, json, דפי office ואף משחק מחשב.

דוגמאות

Aether

שרת בעל גישה עקיפה, שרת זה מתקשר עם שירות רשת SOAP הנכתב בNode.js והועלה לרשת בעזרת Railway. שירות הרשת מציג את הספרים בדף XML חיצוני לצפייה: קישור, הקוד.

Atlas

שרת בעל גישה עקיפה, שרת זה מתקשר עם API שנכתב בפרויקט WebApi. הAPI מאחסן את הספרים בקובץ JSON בפרויקט Storage.

Dummy

שרת מדמה שיוצר ומחזיר נתוני ספר דמה. שרת זה אינו מבצע שום פעולות אמיתיות אבל מספק יישום פשוט למטרות בדיקה.

Echo

שרת בעל גישה ישירה, שרת זה משתמש בDataSet על מנת אחסון In Memory ומאחסן את הספרים בדף XML בפרויקט Storage.

Harmony

שרת בעל גישה עקיפה, שרת זה משתמש בAPI שכתוב בjava ומתקשר עם plugin של משחק מחשב Minecraft על מנת לאחסן ספר ברכיב במשחק המחשב. הקוד, הסבר.

Nimbus 1.0

שרת בעל גישה ישירה, שרת זה מאחסן את מידע הספרים במסד הנתונים SQL Server. במסד נתונים ישנן שתי טבלאות; **BookChapters** פרקי הספר, **BooksMetadata** מידע הספר.

Nimbus 2.0

שרת בעל גישה ישירה, שרת זה הינו הגרסה השנייה של Nimbus. שרת זה משתמש במסד הנתונים MongoDB לעומת SQL Server.

Orion

שרת בעל גישה עקיפה, שרת זה משתמש בשירות רשת בפרויקט WebServices המאחסן את מידע הספרים בדף XML בפרויקט Storage.

Solace

שרת בעל גישה ישירה, שרת זה מאחסן את מידע הספרים באחד מן שלושה קבצים: PowerPoint, Word, Excel (למפתח האפשרות לבחור באיזה קובץ לאחסן את המידע) בפרויקט Storage.

Stegan 1.0

שרת בעל גישה ישירה, שרת זה משתמש בסטגנוגרפיה ומצפין את מידע הספרים לתמונה רנדומלית מAPI חיצוני, לאחר מכן מאחסן את התמונה בתיקייה לוקלית בפרויקט Storage.

Stegan 2.0

שרת בעל גישה ישירה, שרת זה הינו הגרסה השנייה של Stegan. השרת מצפין ומעלה את תמונות הספרים לאינטרנט (הקוד) לבסוף השרת מאחסן את הקישור לתמונה במסד נתונים בפרויקט Storage.

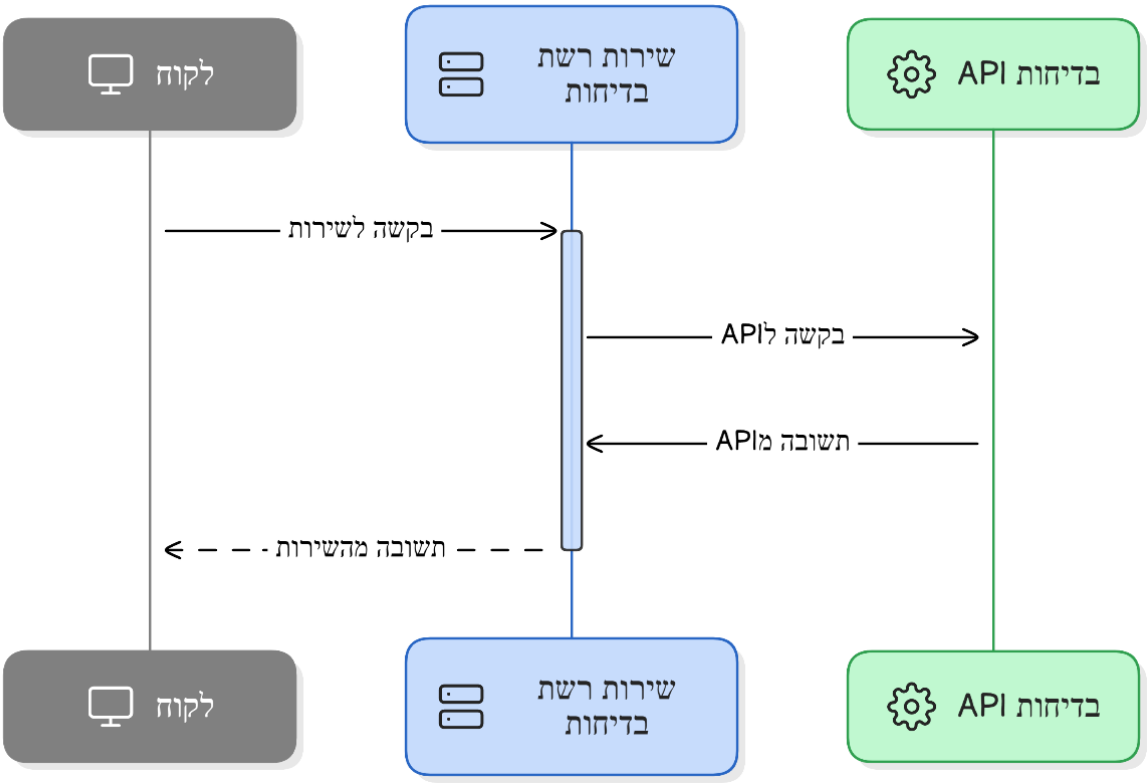
*בתמונה זו מוצפן המידע של אחד הספרים שבאתר.

שירותי רשת

האתר מספק שירות SOAP לניהול ספרים. שירות זה מאפשר פעולות בסיסיות על ספרים כמו הוספה, מחיקה, עדכון וקבלת מידע. שירות רשת זה משומש בשרת Orion.

האתר בנוסף לכך משתמש בשני שירותי רשת נוספים אשר נכתבו בNode.js (הקוד), שירות רשת נוסף לניהול ספרים שמשומש בשרת Aether. ושירות רשת בדיחות.

בשירות רשת הבדיחות ישנן שתי פעולות הוספת מספרים ופעולה המחזירה בדיחה רנדומלית מAPI חיצונית. בנוסף העליתי דף XML מרוחק נוסף המחזיר X בדיחות, לצפייה.



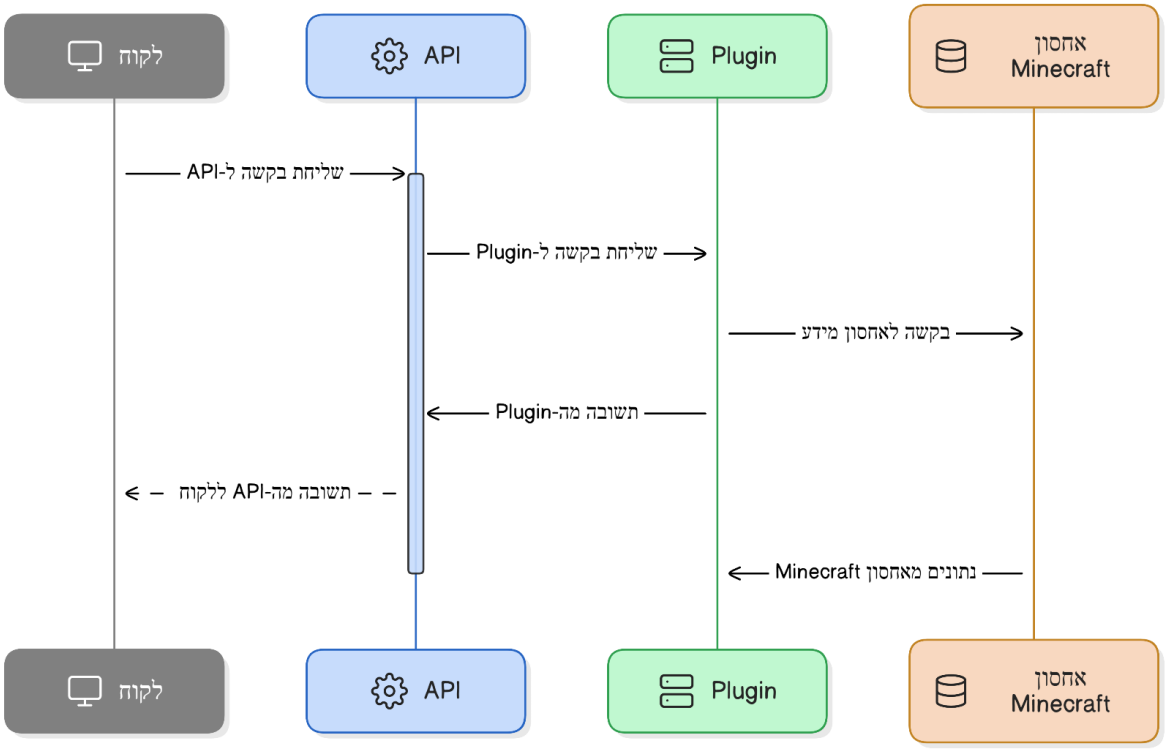
API

האתר מספק שני API, אחד לניהול ספרים שמשומש בשרת Atlas וAPI שני המבצע פעולות פשוטות על ספריות הפרויקט, כגון GET, DELETE, PUT, POST.

בנוסף על כך הפרויקט משתמש בשני API חיצוני שלא כתבתי, אחד לייצור תמונות רנדומליות (קישור) והשני לבדיחות (קישור).

יתרה על כך הפרויקט משתמש בשני API שכתבתי בNode.js להעלאת קבצים ומעקב אחר אירועים באתר נוסף שלי.

ולבסוף הפרויקט משתמש בAPI שנכתב בJAVA לתקשורת עם plugin במשחק המחשב מיינקראפט.



תכנות מונחה עצמים

במערכת נעשה שימוש נרחב בגישה של תכנות מונחה עצמים (OOP) כדי לנהל ולארגן את מבנה הקוד בצורה מודולרית וקריאה.

מחלקות

המערכת כוללת 210 מחלקות אשר משמשות לייצוג של מודלים, שירותים, ומרכיבי ליבה שונים של המערכת. כל מחלקה בנויה באופן שמאפשר לה לבצע תפקיד מסוים במערכת ולספק פונקציונליות ספציפית, תוך שמירה על קפסולציה ושימוש נכון במשתנים ופונקציות.

ממשקים

במערכת ישנם 35 ממשקים (Interfaces) שמספקים חוצץ בין המימושים השונים ומאפשרים יחידות קוד גמישות וניתנות להחלפה. השימוש בממשקים מקדם את עקרון ההפרדה בין הפונקציונליות למימוש שלה, מה שמאפשר לבצע בקלות שינויים ושיפורים במערכת מבלי להשפיע על מרכיבים אחרים.

מונים

במערכת קיימים 7 מונים כדי לייצג ערכים קבועים ומוגדרים מראש, דבר שמאפשר לקוד להיות קריא יותר ומפחית את הסיכון לשגיאות. מונים מאפשרים ייצוג מצבים, סוגים ותתי-סוגים שונים בצורה ברורה וקלה לתחזוקה.

```
1  public enum DocumentType {
2      Word,
3      Excel,
4      PowerPoint,
5  }
```

*מונה עבור טיפוס של קובץ.

מבני נתונים

נעשה שימוש רחב בממשקים גנריים כגון ICollection וIEnumerable במקום סוגים קבועים כגון מערכים או רשימות. הבחירה בממשקים אלה מאפשרת עבודה דינמית וגמישה עם מבני נתונים, שכן הם אינם מחייבים שימוש במבנה נתונים ספציפי ומספקים מגוון רחב של פעולות. (בכל זאת לרוב מתבצעת המרה לList).

תבניות עיצוב

במערכת ישנן תבניות עיצוב רבות בכדי לשפר את ארכיטקטורת המערכת, להוסיף גמישות ולפשט את תהליכי הפיתוח והתחזוקה, אחת מתבניות העיצוב שקיימות היא

מפעל (factory), תבנית עיצוב זו נועדה ליצירת עצמים במערכת בצורה דינמית. לדוגמה, מפעל שרתים היא מחלקה אשר יוצרת שרת לפי הדרישות. באמצעות מפעל ניתן להוסיף סוגי שירותים חדשים למערכת ללא שינוי בקוד הקיים, מה שמוסיף גמישות ומאפשר הרחבה עתידית קלה של המערכת.

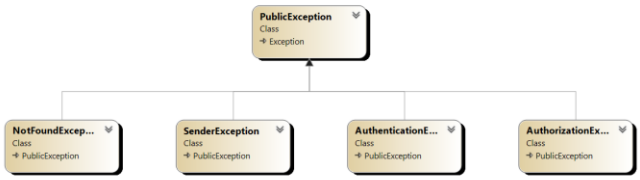
הזרקת תלויות

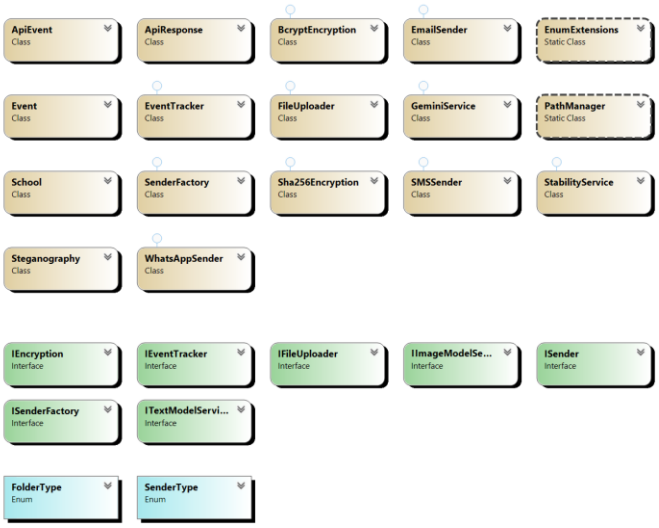
ASP.NET Core מאפשר ניהול יעיל של תלויות בין מחלקות ושירותים שונים במערכת. התלות מוזרקת למחלקות השונות בעת ריצה באמצעות הגדרות ב Program.cs ובמחלקת Startup בשכבת התצוגה של הפרויקט.

דיאגרמת מחלקות

עבור כל פרויקט בasp.net קיימת דיאגרמת מחלקות בעזרת הרחבה בvisual studio 2022.
הדיאגרמה מתארת את מערכת היחסים בין המחלקות שבפרויקט וכוללת מחלקות, ממשקים, מונים וירושה.

*דיאגרמת מחלקות עבור הפרויקט Utils.

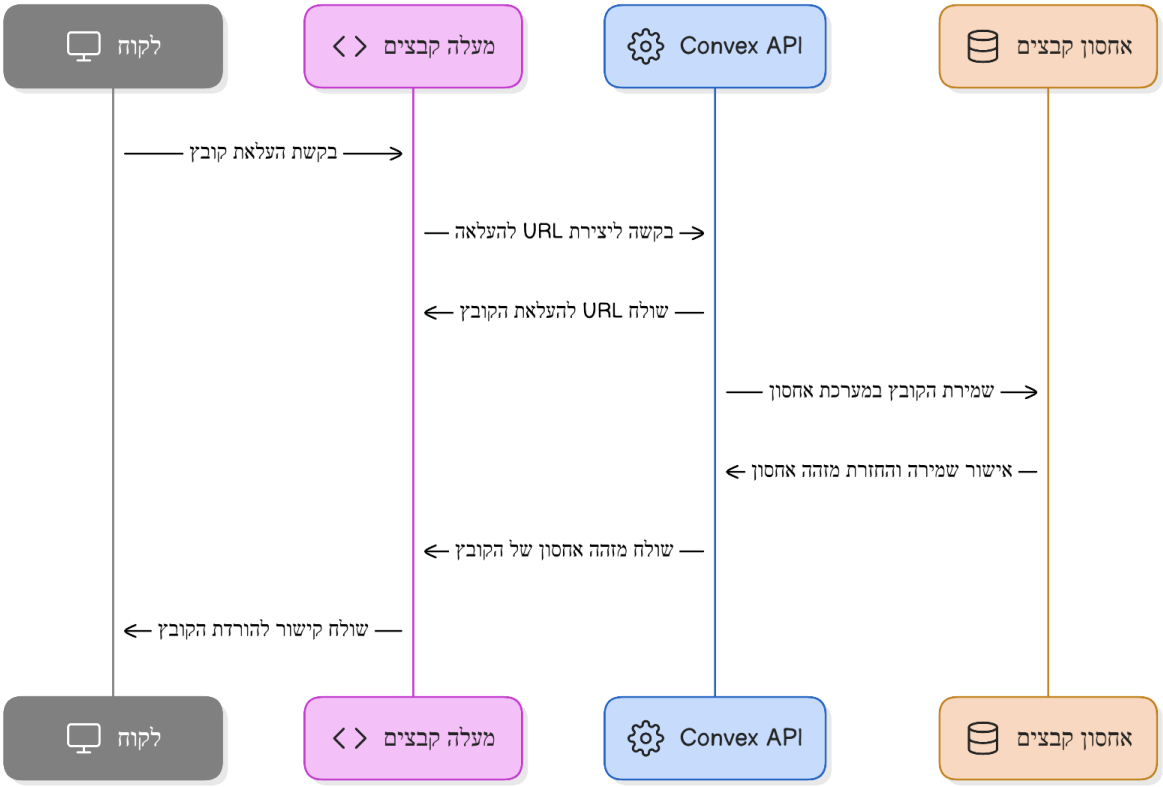




הרחבות

העלאת קבצים

במערכת השתמשתי במסד הנתונים Convex עבור אחסון קבצים. על מנת ליצור קישור להעלאת קובץ וקבלת קישור של קובץ לאחר שמירתו באחסון יצרתי API (הקוד). בנוסף, יצרתי מחלקת העלאת קבצים (FileUploader בפרויקט Utils) שמיועדת לטיפול בקשות מהלקוח. למחלקה זו יש אינטראקציה עם הAPI על מנת לנהל את הקבצים.



טיפול בחריגים

במערכת קיימות מספר מחלקות חריגים היורשות מן מחלקת PublicException.

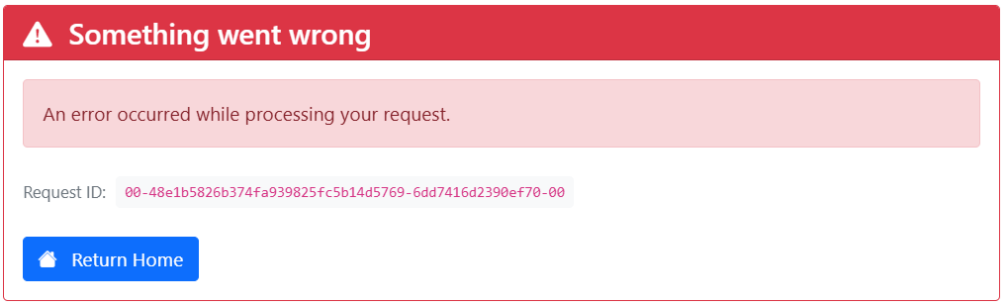
```
1  public class PublicException(string message) : Exception(message) {
2  }
```

שגיאה במערכת עלולה להכיל מידע רגיש או סודי או לא רלוונטי ולכן יש להציג למשתמש אך ורק שגיאות שיורשות מן המחלקה PublicException.

כאשר מתרחשת שגיאה, המשתמש מנותב לעמוד שגיאה. בעמוד זה המערכת חושפת למשתמש את השגיאה אשר התרחשה. כאשר השגיאה יורשת מן PublicException מוצג תוכן השגיאה אחרת מוצג "An error occurred while processing your request".

כך המשתמש מקבל מידע ברור וממוקד, מבלי להיחשף לפרטים טכניים שאינם רלוונטיים או עלולים להכיל מידע רגיש.

שגיאה אשר לא יורשת מן מחלקת PublicException.



שגיאה אשר יורשת מן מחלקת PublicException.

Unit testing

בפרויקט , יצרתי בדיקות יחידה לשירותי הרשת שנכתבו ב - Node כדי לוודא שהם עובדים בצורה תקינה. ישנן בדיקות עבור כל הפעולות שבשירותי הרשת. השתמשתי ב - Supertest כדי לשלוח בקשות לשירות ולוודא שהתגובות נכונות. הבדיקות מבוצעות בעזרת בקשות XML המוגדרות בהתאם לפורמט SOAP כך שהשירות יוכל לקבל את הבקשות ולעבד אותן.

```
1  describe("BooksWebService", () => {
2    it("should return WSDL when requested", async () => {
3    });
4
5    it("should create a book", async () => {
6    });
7
8    it("should get a book", async () => {
9    });
10
11   it("should get all books", async () => {
12   });
13
14   it("should delete a book", async () => {
15   });
16 });
```

הצפנות

במערכת השתמשתי בIdentity User על מנת לנהל משתמשים, הספרייה מטפלת בהצפנת סיסמאות ולכן אין צורך בהצפנה באופן ידני. למרות זאת, כתבתי מחלקות הצפנה הממשות ממשק הצפנה, למחלקות אלו אין תפקיד אך נהגתי בעבר להצפין איתן סיסמאות.

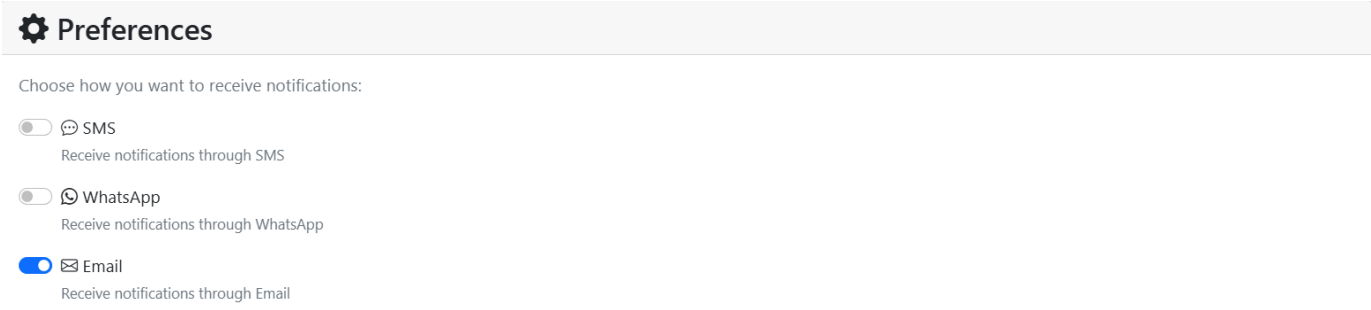
```
1       public interface IEncryption {
2           string Encrypt(string input);
3           bool Compare(string input, string hash);
4       }
5   }
```

בנוסף על כך, במערכת קיימת מחלקת הצפנה של סטגנוגרפיה. המחלקה מאפשרת הצפנה ופתיחה של מידע מתוך תמונות על ידי הסתרה של ביטים של טקסט בתוך הפיקסלים של תמונה. כך ניתן לא רק להסתיר טקסט בתוך תמונה, אלא גם לשלוף את המידע המוסתר במידת הצורך.

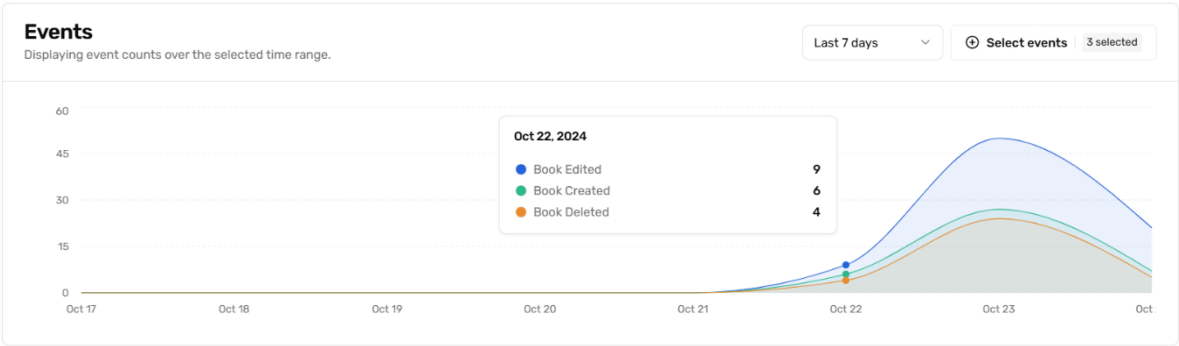
```
1  public class Steganography {
2      public static Bitmap Encode(string text, Bitmap bmp) {
3      }
4
5      public static string Decode(Bitmap bmp) {
6      }
7  }
```

הודעות ואירועים

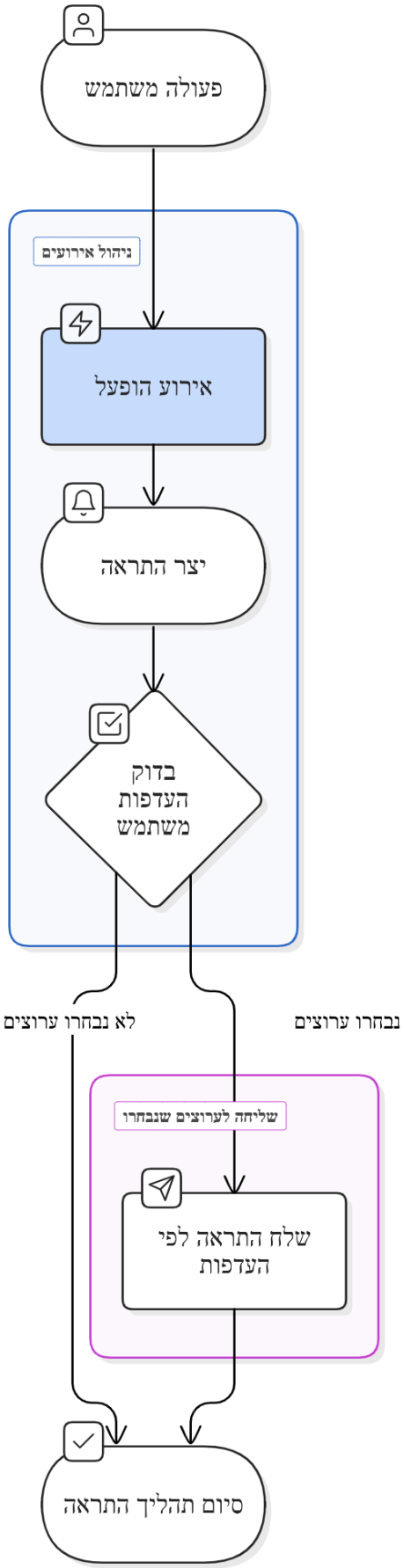
האתר שולח הודעה למשתמש כתוצאה מן אירועים מרובים המרחשים באתר, למשתמש האפשרות לבקש מן האתר לשלוח את הודעות אלה בSMS, WhatsApp וEmail.



ישנן מגבלות לשליחת הודעות בSMS וWhatsApp, האתר אינו מסוגל לשלוח הודעה למספר טלפון שלא אישרתי בספק, כלומר על מנת לשלוח הודעה למספר טלפון אצטרך לאשר אותו לפני כן. על מנת להפטר ממגבלה זו אצטרך לשלם מחיר כספי.

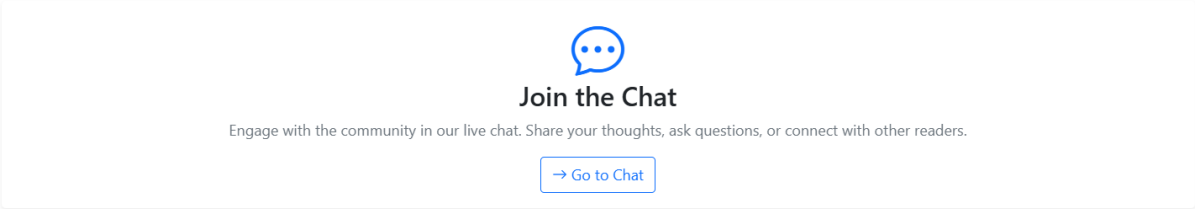


האתר משתמש בAPI חיצוני באתר נוסף שכתבתי (קישור) בו ניתן לראות אנליטיקה על אירועי האתר.

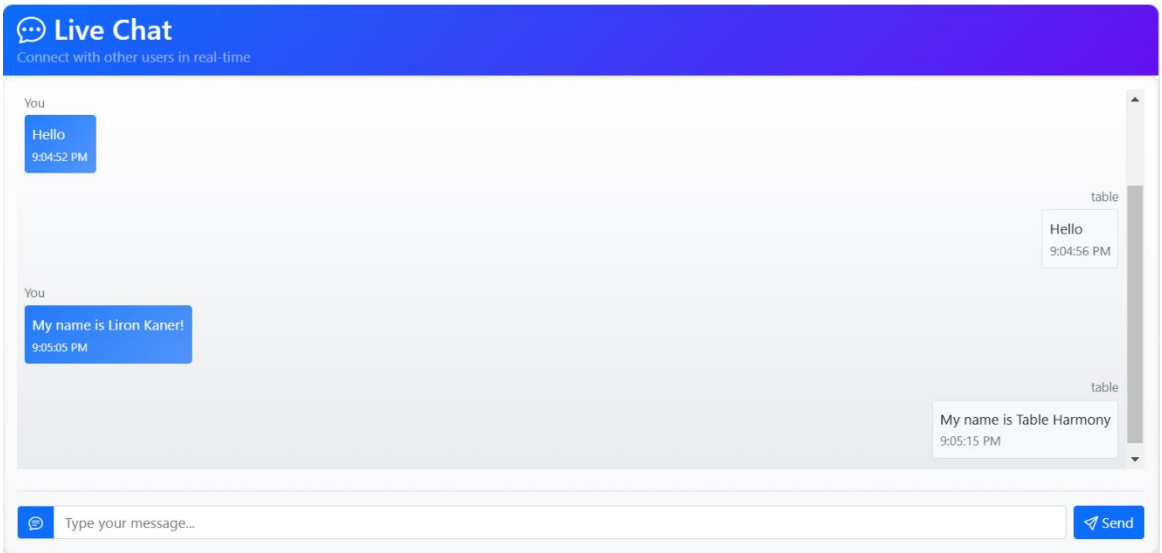

פעולה משתמש
ניהול אירועים
אירוע הופעל
יצר התראה
בדוק העדפות משתמש
לא נבחרו ערוצים
נבחרו ערוצים
שליחה לערוצים שנבחרו
שלח התראה לפי העדפות
סיום תהליך התראה

Live chat

בדף הבית, למשתמשים יש גישה לדף הצ'אט.

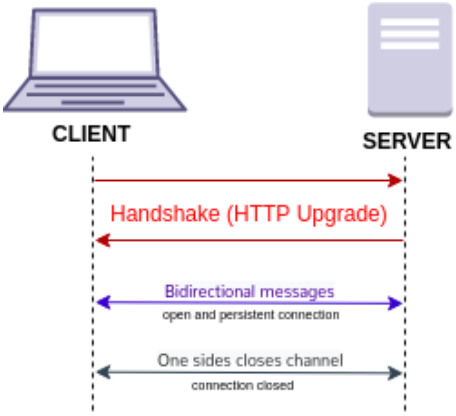


דף הצ'אט הוא דף המאפשר למשתמשים לכתוב הודעות ולצפות בהודעות מכלל המשתתפים בזמן אמת.



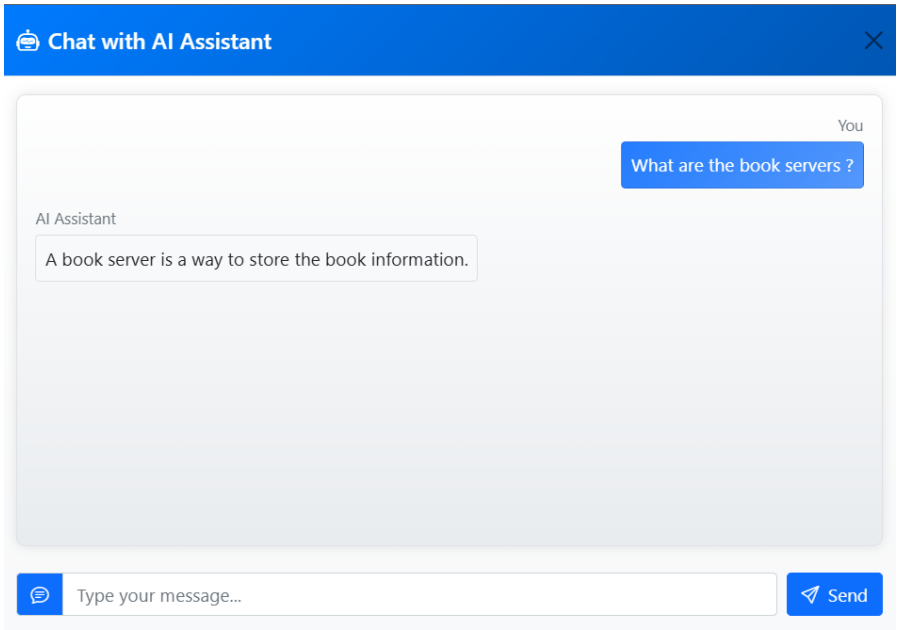
דף הצ'אט פועל לפי פרוטוקול Web Socket לתקשורת בין שרת ולקוח בזמן אמת.

השרת נכתב בפרויקט WebSockets בעזרת ספריית SignalR, בעוד והלקוח נכתב בפרויקט הPresentation המספק ממשק משתמש לצ'אט.



בינה מלאכותית

בדף הבית, למשתמשים יש גישה לAi Assistant (סרטון תצוגה).



לתמיכה טכנית או שאלות, משתמשים נעזרים בעוזר הבינה המלאכותית אשר מסוגל לענות על היבטים רבים של האתר. הבינה המלאכותית היא הLLM של Google, Gemini Flash.

בפרויקט Utils, הממשק למען שירות בינה מלאכותית.

```
1  public interface ITextModelService {
2      Task<string> GetResponseAsync(string prompt);
3  }
```

המחלקה GeminiService, משתמשת במודל Gemini Flash וממשת את הממשק למען שירות בינה מלאכותית (עבור שימוש GeminiService מוזרק בפרויקט התצוגה).

```
1  public class GeminiService : ITextModelService {
2      private readonly GenerativeModel _model;
3
4      public GeminiService(IConfiguration configuration) {
5      }
6
7      public async Task<string> GetResponseAsync(string prompt) {
8      }
9  }
```

בנוסף בפרויקט Utils, הממשק למען בינה מלאכותית נוסף עבור מודלים יצרניים.

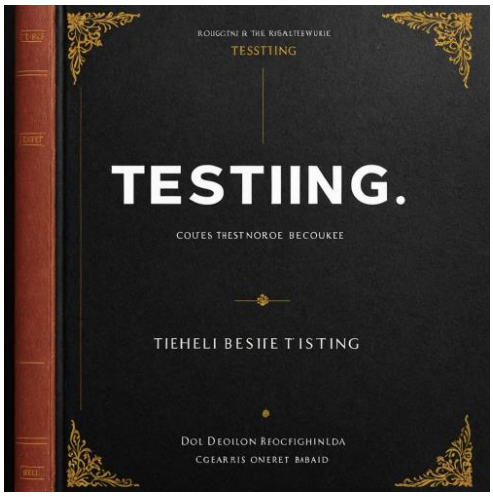
```
public interface IImageModelService {
    public readonly record struct Image(string Prompt, int Width = 1000, int Height = 1500, string Format = "webp");

    Task<Stream> GenerateImageAsync(Image image);
}
```

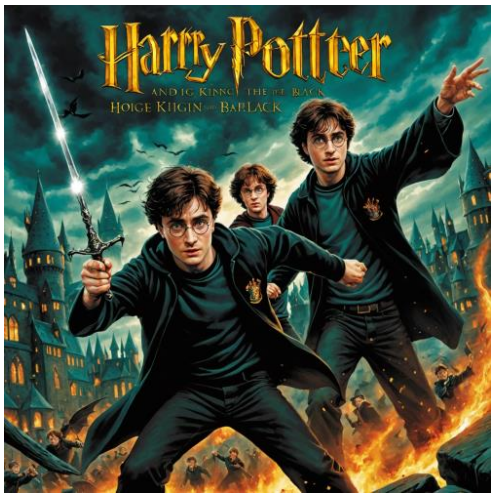
המחלקה StabilityService, משתמשת Stable Diffusion וממשת את הממשק למען שירות בינה מלאכותית (עבור שימוש StabilityService מוזרק בפרויקט התצוגה).

```
public class StabilityService : IImageModelService {
    private readonly HttpClient _httpClient;

    public StabilityService(IConfiguration configuration) {
    }

    public async Task<Stream> GenerateImageAsync(IImageModelService.Image image) {
    }
}

```

במערכת, כאשר משתמש עורך או יוצר ספר המשתמש מסוגל לבקש מן הבינה המלאכותית ליצור כריכה עבור הספר, הכריכה תיוצר לפי הכותרת והתיאור של הספר. לאחר שהבינה המלאכותית תיצור את התמונה המערכת מעלה את התמונה בעזרת מחלקת העלאת הקבצים.

דוגמאות לכריכות של ספרים שיוצרו על ידי הבינה המלאכותית:





רפלקציה

במהלך הפרויקט, נהניתי מאוד מתהליך הבנייה והפיתוח שלו, והרגשתי שזה מאפשר לי להיות יצירתי ולהתמודד עם אתגרים חדשים ומעניינים. למדתי כיצד להיות יצירתי ולממש את הרעיונות שלי בצורה יעילה יותר. בסופו של דבר, החוויה הייתה מאתגרת ומרתקת והעניקה לי מגוון יכולות חדשות. אני נהניתי מהתהליך והבנתי שהלמידה והפיתוח הם תהליכים רציניים ומאתגרים אך גם מאפשרים לי להיות יצירתי ומלמדים אותי להתמודד עם קשיים ולמצוא פתרונות חדשים ויצירתיים.

אמצעי כניסה

לצורך כניסה בתור משתמש רגיל, יש למלא את הפרטים הבאים:

אמייל: normal@email.com

סיסמה: Normal123!

לצורך כניסה בתור משתמש מנהל, יש למלא את הפרטים הבאים:

אמייל admin@email.com

סיסמה: Admin123!

על מנת להריץ את הפרויקט יש להשתמש בvisual studio 2022 בגרסה +17.8 מכיוון והפרויקט משתמש ב8 net.

נספחים

סרטון הסבר על האתר.

API העלאת קבצים.

API מעקב אירועים.

API/Plugin לתקשורת עם משחק המחשב Minecraft.

שירותי רשת בNode.js.

ביבליוגרפיה

ASP.NET Core – מסגרת ליצירת אתרים.

Entity Framework – ORM.

Twilio – שליחת הודעות.

MailerSend – שליחת אימיילים.

Syncfusion – ניהול מצגת.