מסמך תכנון ועיצוב

מערכת לניהול תיקי מניות

Invest Insight

שם: רועי דניאל
ת"ז: 315707273

תוכן העניינים

1. תיאור המערכת

1.1. כללי

המערכת בנויה מ2 שכבות.

- שכבת השרת (Backend) – אחראית על ביצוע כל הלוגיקה הדרושה, אינטראקציה עם בסיס הנתונים, ומספקת ממשק פשוט לביצוע פעולות בשרת.
- שכבת ממשק המשתמש (Frontend) – אחראית על חוויית המשתמש והצגת המידע שהתקבל מהשרת.

התקשורת בין חלקי המערכת הינה באמצעות בקשות בפרוטוקול Http.
כך לדוגמה בעת התחברות המשתמש, נשלחת בקשה לשרת דרך ממשק המשתמש (Frontend) עם פרטי ההתחברות (אימייל וסיסמה).
השרת (Backend) מבצע בדיקה בבסיס הנתונים כדי לוודא שהפרטים נכונים.
לאחר הבדיקה, השרת מחזיר תגובה לממשק המשתמש, הכוללת מידע על תקינות ההתחברות (לדוגמה, האם ההתחברות הצליחה או נכשלת).
אם ההתחברות הצליחה, המשתמש מועבר לניהול התיקים שלו ופרטי ההתחברות נשמרים בדפדפן.

1.2. הנחות עבודה בכתיבת הפרויקט

1. משתמש לא יכול להתחבר במקביל ממספר מחשבים.
2. המערכת אינה מאפשרת קבלת מידע בזמן אמת אלא בפרקים כל חצי שעה מעת פתיחת המסחר ועד לסגירתו.
3. המערכת אינה מאפשרת קבלת מידע על שינויים בשוק ההון לאחר סגירת מסחר או לפני פתיחתו.
4. המערכת שומרת נתונים על מניות שהתבקשה לקבל עליהם מידע בלבד (בקשה ספציפית למניה לפי סימול).
5. תחזיות לערך מניה עתידי מעודכנות אחת למספר ימים.
6. מניה חדשה נוספת למאגר המידע ללא תחזית ולכן תחזיות יעודכנו במהלך הימים העוקבים.
7. המניות הנשמרות באתר יהיו מניות מהשוק האמריקאי בלבד.
8. בבסיס הנתונים ישמרו נתונים על מניות בלבד ולא קרנות סל.
9. לא מתבצע אימות המשתמש בעת שחזור סיסמה.

1.3. מוסכמות

1. מושגים:
    1. Stock - מניה
    2. Share - חלקי המניה (כמות המניות ממניה מסוימת)
    3. Trend - מגמה (לדוגמה מניות בעליות, מניות בירידות, מניות פעילות)
2. רישום:
    1. מייל יהיה רשום באופן הבא: email@mailprovider.domain.
3. שמות מחלקות ומשתנים:
    1. בשמות קבצים כל מילה מתחילה באות גדולה וכל מילה אחריה גם כן.
    2. שמות משתנים מתחילים באות קטנה ולאחר מכן כל מילה נוספת באות גדולה (camelCase).
    3. משתנים קבועים באותיות גדולות ובין מילים קיים "_".
    4. שמות פונקציות בשרת מתחילות באות גדולה ולאחר מכן כל מילה באות גדולה (PascalCase).
    5. שמות פונקציות בממשק משתמש מתחילות באות קטנה ולאחר מכן כל מילה נוספת באות גדולה (camelCase).
    6. שמות של פונקציות א-סינכרוניות נגמר תמיד בAsync (צד שרת בלבד) על מנת לאפשר לנו להבדיל בין פעולות סינכרוניות וא-סינכרוניות.

2. שרת

2.1. כללי

פרוייקט זה מכיל את הממשק לשרת, נבצע הפרדה בין פרוייקט הריצה לפרוייקטי עזר שבהם נעשה שימוש.

השרת מחולק ל3 שכבות:

1. StocksApi - ממשק השרת.
   מקבל את כל הבקשות, מבצע עיבוד, גישה לבסיס הנתונים ועדכון ערכי מניה ממערכות finance.
2. Library - ספריית מחלקות משותפות לפרוייקט StocksApi וYahooFinance.
3. YahooFinance - ספריית ניהול נתוני yahoo finance.
   מאפשרת קבלת ערכי מניה, חיפוש מניות, קבלת טרנדים והמלצות מאתרי finance השונים.

2.2. עקרונות מנחים

- הפרדה בין פרוייקט הריצה לבין פרוייקט נתוני YahooFinance.
- יש לשמור על מידע ככל הניתן כדי לחסוך צורך ביצירת בקשות להבאת נתונים בזמן שהמשתמש מבקש.
- יש לחסום פעולות לא תקינות במערכת כמו ניסיון להוסיף מניות לתיק שאינו קיים וכו'.

StocksApi .2.3

מסד הנתונים .2.3.1

מסד הנתונים של האתר מבוסס על mongodb ולכן מבוסס על מחלקות הקיימות במודלים שלנו מהם נוצרות רשימות והן:

1. User.
2. Stock.
3. MarketTrend.
4. SearchResult.

בסיס הנתונים אינו רלציוני ולכן משתמש במסמכים על מנת לשמור על הנתונים.

2.3.2. ממשקים

- IUsersDal.cs

ממשק המאפשר גישה למאגר הנתונים על משתמשים, מאפשר לבצע את כל הפעולות הנדרשות למשתמש, בין היתר להוסיף רשימות, להוסיף מניות לרשימה וכו'.

```
Task<List<User>> FindAllAsync();
Task<User?> FindOneByIdAsync(string id);
Task CreateAsync(User user);
Task UpdateAsync(string id, User updatedUser);
Task RemoveAsync(string id);
Task<User?> FindOneByEmailAsync(string email);
Task UpdatePasswordAsync(string email, string updatedPassword);
Task AddNotificationAsync(string userEmail, StockNotification notification);
Task AddStockListAsync(string userEmail, string listName);
Task RemoveStockListAsync(string email, string listName);
Task NotifyUserAsync(string userEmail, string notificationId);
Task RemoveNotificationAsync(string userId, string notificationId);
```

- IStocksDal.cs

ממשק המאפשר גישה למאגר הנתונים על מניות, בין היתר מאפשר עדכון, יצירה והוספת התראות ועוד.

```
Task<List<Stock>> FindAllAsync();
Task<Stock?> FindByIdAsync(string id);
Task<Stock?> FindBySymbolAsync(string symbol);
Task CreateAsync(Stock stock);
Task UpdateAsync(string id, Stock updatedStock);
Task RemoveAsync(string id);
Task UpdateAnalysisBulkAsync(List<StockAnalysis> stocksAnalysis);
Task UpdateStockPriceBulkAsync(List<PriceResponse> stockPriceResponses);
Task<List<Stock>> FindManyBySymbolAsync(string[] symbols);
Task AddNotificationAsync(StockNotification stockNotification);
Task RemoveNotificationAsync(string symbol, string notificationId);
```

- ITrendsDal.cs

הממשק למאגר הנתונים של הטרנדים הינו פשוט יותר היות ואנחנו רוצים ליצור בתחילה את הטרנדים ולאחר מכן נצטרך רק לעדכן במידה ויש עדכון בטרנד מסוים.

```
Task CreateAsync(MarketTrend trend);
Task<List<MarketTrend>> FindAsync();
Task<MarketTrend?> FindOneAsync(string trendName);
Task RemoveAsync(string trendName);
Task AddOrUpdateOneAsync(string trendName, MarketTrend marketTrend);
```

- ISearchResultsDal.cs

ממשק למאגר מידע של חיפוש הינו הפשוט ביותר היות והחיפוש נשמר כמו שהוא ולכן יש בדיקה האם החיפוש בוצע בעבר ואם לא מתבצע חיפוש ונשמר במאגר.

```
Task CreateAsync(SearchResult item);
Task<SearchResult?> FindBySearchTermAsync(string searchTerm);
```

- ISearchResultRepository.cs

ממשק זה מאפשר למשתמש לבקש תוצאות חיפוש למניות.

```
Task<List<StockSearchResult>> SearchStockByTermAsync(string searchTerm);
```

- IStockRepository.cs

ממשק לביצוע פעולות הקשורות למניות, בין היתר קבלת נתוני מניה, הוספת התראות וכו׳.

```
Task<List<Stock>> GetAllAsync();
Task<Stock?> GetStockBySymbolAsync(string symbol);
Task<List<Stock>> GetStocksBySymbolAsync(string[] symbol);
Task UpdateStocksBySymbolAsync(string[] stockSymbols);
Task UpdateStocksAnalysisAsync(string[] orderedStockSymbols);
Task RemoveNotificationAsync(string symbol, string notificationId);
Task AddNotificationAsync(StockNotification stockNotification);
```

- ITrendRepository.cs

ממשק לקבלת תוצאות טרנדים.

```
Task<List<MarketTrend>> GetMarketTrendsAsync();
Task<MarketTrend?> GetTrendAsync(string trendType);
```

- IUserRepository.cs

ממשק לקבלת מידע על משתמשים, עדכון משתמשים וניהול תיקי המניות של המשתמשים.

```
Task AddNotificationAsync(StockNotification notification);
Task<Share> AddShareAsync(string id, SharePurchase share);
Task AddStockListAsync(StockListDetails stockListDetails);
Task AddUserAsync(UserDetails userDetails);
Task AddWatchingStockAsync(string id, string listName, string stockSymbol);
Task<bool> ConnectUserAsync(UserCredentials user);
Task RemoveNotificationAsync(string id, string notificationId);
Task<List<User>> GetAllAsync();
Task<User?> GetAsync(string email);
Task RemoveShareAsync(ShareSale shareSale);
Task RemoveStockListAsync(StockListDetails stockListDetails);
Task RemoveUserAsync(string id);
Task RemoveWatchingStockAsync(string id, string listName, string stockSymbol);
Task ShowNotificationAsync(StockNotification notification);
Task UpdateShareNoteAsync(WatchingStockAction watchingStockAction);
Task UpdatePasswordAsync(PasswordUpdateRequest passwordUpdateRequest);
```

- IPasswordHasher.cs

ממשק לביצוע הצפנה של סיסמאות באמצעות hashing.

```
string HashPassword(string password);
bool VerifyPassword(string password, string hashedPassword);
```

- IStockMarketTime.cs

ממשק לקבלת מידע אודות פתיחת המסחר בבורסה בארה"ב.

```
bool IsMarketOpen(DateTime date);
bool ShouldStockBeUpdated(Stock stock);
```

- IStockNotificationSender.cs

ממשק לשליחת התראות למשתמש, ממשק זה משנה התראה שאינה מופעלת להתראה פעילה במידה ומתקיימים כל התנאים.

```
Task HandleStockPriceUpdatesAsync(params string[] stockSymbols);
```

2.3.3. מחלקות

- יוצרים (Generator)

אנחנו משתמשים במחלקות יוצרים כדי למלא ערכים חסרים בעת קבלת מידע בין אם מהמשתמש או ערכי מניות וכו'.
כלל מחלקות היוצרים שלנו הינם מחלקות סטטיות ולכן אינן ממשות ממשק.

SearchResultGenerator.cs

```
static SearchResult Generate(string searchTerm,
        List<StockSearchResult> stockSearchResults);
```

ShareGenerator.cs

```
public static Share Generate(SharePurchase sharePurchase);
```

StockGenerator.cs

```
public static Stock? Generate(PriceResponse? yahooStock);
```

TrendGenerator.cs

```
public static MarketTrend Generate(string trendName, MarketTrendsResponse
        marketTrendsResponse);
```

UserGenerator.cs

```
public static User Generate(UserDetails userDetails);
```

WatchingStockGenerator.cs

```
public static WatchingStock Generate(string stockSymbol);
```

- Controllers

קונטרולרים הם הממשק החיצוני שניתן מול השרת שלנו והדרך לגשת לבסיס הנתונים.
יורש ממחלקה מופשטת Controller המאפשרת קבלת פניות Http לשרת.
עברו מחלקות אלה נרשום את נתיב הגישה הכללי ולכל פונקציה ניתן סוג הבקשה ונתיב פנימי במידה וקיים.

ShareController.cs

מחלקה לקבלת פניות הקשורות לניהול תיקי ההשקעות של משתמשים, בין היתר מאפשרת הוספת ומחיקה של מניות, חלקי מניות מהתיקים השונים.
המחלקה משתמשת במימוש של הממשקים הבאים:

1. IUserRepository - קבלת נתונים על משתמשים ועדכון פרטים על תיקי מניות.
2. IStockRepository - בדיקה האם מניה קיימת לפי סימול בעת הוספת רכישה לתיק.

נתיב הגישה הכללי: /api/Share

```
[HttpPost] Task<IActionResult> AddShareAsync([FromBody] SharePurchase
sharePurchase)

[HttpDelete] Task<IActionResult> RemoveShareAsync(
        [FromBody] ShareSale shareSale)

[HttpPost("list")] Task<IActionResult> AddUserListAsync([FromBody]
StockListDetails stockListDetails)

[HttpDelete("list")] Task<IActionResult> RemoveUserListAsync([FromBody]
StockListDetails stockListDetails)

[HttpPost("watching-stock")] Task<IActionResult> AddWatchingStockAsync(
        [FromBody] WatchingStockAction watchingStockAction)

[HttpDelete("watching-stock")] Task<IActionResult>
        UpdateWatchingStockNoteAsync([FromBody] WatchingStockAction
        watchingStockAction)

[HttpPatch("watching-stock-note")] Task<IActionResult>
        RemoveWatchingStockAsync([FromBody] WatchingStockAction
        watchingStockAction)
```

StockController.cs

מחלקה לקבלת פניות הקשורות לערכי המניות וטרנדים של מניות.

המחלקה משתמשת במימוש של הממשקים הבאים:

1. IStockRepository - לצורך קבלת נתוני מניות ועדכון ערכי מניות.
2. ITrendRepository - לצורך קבלת נתונים על טרנדים.
3. ISearchResultRepository - לצורך קבלת נתונים על חיפוש מניות לפי מונחים.

נתיב הגישה הכללי: /api/Stock

```
[HttpGet] Task<List<Stock>> GetAllStocksAsync()

[HttpGet("symbol/{symbol}")] Task<ActionResult<Stock>>
        GetStockBySymbolAsync(string symbol)

[HttpPost("symbol/bulk")] Task<IActionResult> GetStocksBySymbolsAsync(
        string[] symbols)

[HttpGet("find/{searchTerm}")] Task<ActionResult<List<StockSearchResult>>>
        FindSymbolByTermAsync(string searchTerm)

[HttpGet("marketTrends")] Task<ActionResult<List<MarketTrend>>>
        GetMarketTrendsAsync()

[HttpPost("force-update-all-stocks")] Task<IActionResult>
        ForceUpdateAllStocksAsync()
```

UserController.cs

מחלקה לקבלת פניות הקשורות למשתמשים, מאפשרת לקבל מידע על משתמש, מחיקה ויצירה של משתמש ומחיקת התראות.

המחלקה משתמשת במימוש של הממשקים הבאים:

1. IStockRepository - לצורך קבלת נתונים על מניה והוספת התראות למניות.
2. IUserRepository - לצורך קבלת נתונים על משתמשים וביצוע פעולות נוספות כמו מחיקה, יצירה, עדכון.

נתיב הגישה הכללי: /api/User

```
[HttpGet] Task<ActionResult<User>> GetByEmailAsync(
        [FromQuery] string email)

[HttpDelete] Task<IActionResult> DeleteAsync([FromQuery] string email)

[HttpGet(“all")] Task<List<User>> GetAsync()

[HttpPost(“connect-user")] Task<IActionResult> ConnectUserAsync(
        [FromBody] UserCredentials user)

[HttpPost(“register")] Task<IActionResult> CreateAsync(UserDetails userDetails)

[HttpPost(“update-password")] Task<IActionResult>
        UpdatePasswordAsync(PasswordUpdateRequest passwordUpdateRequest)

[HttpPost(“notification")] Task<IActionResult>
        AddStockNotificationAsync(StockNotification stockNotification)

[HttpDelete(“notification")] Task<IActionResult>
        DeleteNotificationAsync([FromQuery] string email,
        [FromQuery] string notificationId)
```

- Repositories

מחלקות אלו מאפשרות גישה מוגבלת לבסיס הנתונים.
קיימת מחלקה לכל Dal שקיים על מנת שנוכל לבצע פעולות רלוונטיות.

SearchResultRepository

המחלקה מממשת את הממשק ISearchResultRepository.
המחלקה מקבלת כפרמטרים מימושים של הממשקים הבאים:

1. ISearchResultsDal - לצורך חיפוש ויצירת ערך החיפוש שהתבקש על ידי המשתמש.
2. IFinanceStrategy - לצורך ביצוע חיפוש מניות בשרת חיצוני המספק נתונים אלה.

StockRepository

המחלקה מממשת את הממשק IStockRepository.
המחלקה מקבלת כפרמטרים מימושים של הממשקים הבאים:

1. IStocksDal - לצורך ביצוע פעולות הנדרשות במניות, בין היתר עדכון ערכי מניה והחזרת נתונים על מניות.
2. IFinanceStrategy - לצורך ביצוע חיפוש עדכון ערכי מניה מהבורסה.

TrendRepository

המחלקה מממשת את הממשק ITrendRepository.
מחזירה את ערך הטרנד ומעדכנת אחת ליומיים את הערכים במידה וניתן לעדכן.
המחלקה מקבלת כפרמטרים מימושים של הממשקים הבאים:

1. ITrendsDal - לצורך חיפוש ועדכון הטרנדים.
2. IRealTimeFinanceData - לצורך ביצוע חיפוש ערכי הטרנדים.
3. IAppConfiguration - לצורך קבלת שמות הטרנדים הנדרשים לקבלת נתונים על ידי קובץ הקונפיגורציות.

UserRepository

המחלקה מממשת את הממשק IUserRepository.
המחלקה מקבלת כפרמטרים מימושים של הממשקים הבאים:

1. IUsersDal - לצורך קבלת נתוני משתמשים, עדכון ומחיקה.
2. IPasswordHasher - לצורך ביצוע הצפנה בעת הרשמה.
3. IStockRepository - לצורך חיפוש מניות, מוודא כי הן קיימות בעת הוספת מניות לתיק.

- Services

AppConfiguration

מחלקה זו מממשת את הממשק IAppConfiguration הקיים במחלקה המשותפת Library.

המחלקה מקבלת כפרמטר מימוש של הממשק IConfiguration (מובנה של DotNet), בעזרתה יכולה לגשת לקובץ הקונפיגורציה שמוגדר.

PasswordHasher

מחלקה זו מממשת את הממשק IPasswordHasher.

המחלקה משתמשת במימוש של IAppConfiguration על מנת שתוכל לקבל משתנה לצורך ביצוע hashing.

StockAnalysisUpdater

מחלקה זו מממשת את הממשק BackgroundService, ממשק של Microsoft המאפשר עבודה ברקע באופן אסינכרוני מנוהל על ידי המערכת.

המחלקה משתמשת במספר מימושים:

1. IStockRepository - לצורך קבלת ערכי המניות ועדכונם במידת הצורך.
2. IAppConfiguration - לצורך קבלת משתנה למספר הפעמים שניתן לשלוח בקשה לעדכון תחזית לערך מניה.

StockMarketTime

מחלקה זו מממשת את הממשק IStockMarketTime.

המחלקה משתמשת במימוש של IAppConfiguration על מנת שתוכל לקבל משתנים קבועים שהוגדרו מראש בקובץ הקונפיגורציות, בין היתר זמני פתיחה וסגירה של הבורסה ועוד.

StockNotificationSender

מחלקה זו מממשת את הממשק IStockNotificationSender.

המחלקה משתמשת במימוש של הממשקים IUserRepository וIStockRepository כדי לשלוח התראות למשתמשים עקב הגעה למחיר יעד שנקבע.

StockAutomaticUpdater

מחלקה זו מממשת את הממשק BackgroundService, ממשק של Microsoft המאפשר עבודה ברקע באופן אסינכרוני מנוהל על ידי המערכת.

המחלקה משתמשת במספר מימושים:

1. IStockRepository - לצורך עדכון ערכי המניות.
2. IStockMarketTime - לבדיקה האם הבורסה נפתחה ויש להתחיל עדכון ערכי מניות.
3. IStockNotificationSender - לאחר עדכון ערכי המניות הערכים החדשים נשלחים לשולח ההתראות ובמידה ונדרש ישלח עדכונים למשתמשים.

- Dal

מחלקות אשר יכולות להשתמש ישירות בבסיס הנתונים לפי מודל הרלוונטי למחלקה.
כלל המחלקות משתמשות במימוש של המחלקה IMongoCollection לפי מודל שיוצג לכל מחלקה.

SearchResultsDal
המחלקה מנהלת את בסיס הנתונים ששומר על SearchResult.
המחלקה מממשת את הממשק ISearchResultsDal ומשתמשת במימוש של מחלקה IMongoCollection.

StocksDal
מחלקה המנהלת את בסיס הנתונים ששומר על נתוני המניות Stock.
המחלקה יורשת מהממשק IStocksDal עבור המודל Stock.

UsersDal
מחלקה המנהלת את בסיס הנתונים ששומר על נתוני משתמשים User.
גם מחלקה זו יורשת מהממשק IUsersDal עבור המודל User.

TrendsDal
מחלקה המנהלת את בסיס הנתונים השומר על נתוני הטרנדים באתר Trend.
מחלקה זו מממשת ממשק ITrendsDal.

2.4. Library

2.4.1. מודלים כלליים

- variables.cs

מחלקה זו מכילה את כל המשתנים של השרת, לשימוש כללי של האתר בין היתר שעות פתיחה של הבורסה.

```
public required string PASSWORD_SALT { get; set; }
public int OPEN_MARKET_DAY { get; set; }
public int CLOSED_MARKET_DAY { get; set; }
public int OPEN_MARKET_HOURS { get; set; }
public int CLOSED_MARKET_HOURS { get; set; }
public int OPEN_MARKET_MINUTES { get; set; }
public int MINUTE_INTERVAL_BETWEEN_UPDATE { get; set; }
public int MAX_ALLOWED_ANALYSIS_PER_DAY { get; set; }
```

- ConfigurationKeys.cs

מחלקה זו מכילה את כל שמות הקונפיגורציה הרלוונטיים לשרת, כדי שנוכל לגשת אליהם דרך קובץ הקונפיגורציות בצורה נוחה ומוגדרת.

בין היתר קיים משתנה עבור קונפיגורציה של ConnectionString לחיבור למונגו וזהו משתנה שנרצה לשמור בקובץ נפרד מקובץ הקונפיגורציה ונכניס בצורה עקיפה בקבצים שאינם נגישים ליוזרים חיצוניים כדי לשמור על בטחון המידע שלנו.

```
public const string ConnectionStringSection = "CONNECTION_STRING";
public const string AppVariablesSection = "AppVariables";
public const string DatabaseSettingsSection = "DatabaseSettings";
public const string Finance15Section = "YahooFinance15";
public const string Finance127Section = "YahooFinance127";
public const string Finance1Section = "YahooFinance1";
public const string RealTimeFinanceDataSection = "RealTimeFinanceData";
public const string RealTimeFinanceMarketTrends = "MarketTrends";
```

- DatabaseSettings.cs

מחלקה זו מכילה את כל הפרטים הרלוונטיים להתחברות לבסיס הנתונים שלנו (mongo db) למעט connection string אשר מוגדר באמצעות ConfigurationKeys.

```
public required string DatabaseName { get; set; }
public required string StocksCollectionName { get; set; }
public required string UsersCollectionName { get; set; }
public required string MarketTrendsCollectionName { get; set; }
public required string SearchResultsCollectionName { get; set; }
```

2.4.2. מודלים למשתמש (users)

- User.cs

מחלקה זו מכילה את כל המשתנים הרלוונטיים לשמירת פרטי משתמש, פרטי מניות, תיקי מניות ופרטי התחברות של המשתמש.

```
public required string Id { get; set; }
public required string FirstName { get; set; }
public required string LastName { get; set; }
public required string Email { get; set; }
public required string Password { get; set; }
public required Dictionary<string, Dictionary<string, WatchingStock>>
        WatchingStocksByListName { get; set; }
```

- UserCredentials

מחלקה זו מכילה את המשתנים הנדרשים לצורך התחברות משתמש לאתר.

```
public required string Email { get; set; }
public required string Password { get; set; }
```

- UserDetails.cs

מחלקה זו מכילה את כל המשתנים הנדרשים לצורך הרשמת משתמש חדש לאתר.

```
public required string FirstName { get; set; }
public required string LastName { get; set; }
public required string Email { get; set; }
```

```
public required string Password { get; set; }
```

- PasswordUpdateRequest

מחלקה המכילה פרטים לצורך ביצוע עדכון סיסמה של משתמש.

```
public required string Email { get; set; }
public required string Password { get; set; }
```

- StockNotification

מחלקה המכילה פרטים הרלוונטיים לשמירת פרטי התראות למחירי מניה.

```
public string? Id { get; set; }
public required string StockSymbol { get; set; }
public required string UserEmail { get; set; }
public required double TargetPrice { get; set; }
public bool IsTargetBiggerThanOrEqual { get; set; }
public bool ShouldBeNotified { get; set; }
```

2.4.3. מודלים למניות (stock)

- Stock.cs

מחלקה זו מכילה את כל הנתונים הרלוונטיים לאתר על מניות, בין היתר ערך המניה, שם וכו'.

```
public required string Id { get; set; }
public required string Name { get; set; }
public required string Symbol { get; set; }
public required double Price { get; set; }
public required double FiftyDayAverage { get; set; }
public required double TwoHundredDayAverage { get; set; }
public required string FiftyTwoWeekRange { get; set; }
public required double FiftyTwoWeekLow { get; set; }
public required double FiftyTwoWeekHigh { get; set; }
public required DateTime UpdatedTime { get; set; }
public string? AnalystRating { get; set; }
public StockAnalysis? Analysis { get; set; }
public required List<StockNotification> StockNotifications { get; set; }
```

- StockAnalysis.cs

מחלקה זו מכילה את כל המשתנים הרלוונטיים לשמירת תחזיות לערכי מניה עתידיים.

```
public required string Symbol { get; set; }
public DateTime UpdatedTime { get; set; }
public double TargetHighPrice { get; set; }
public double TargetLowPrice { get; set; }
public double TargetMeanPrice { get; set; }
public double TargetMedianPrice { get; set; }
```

2.4.4. מודלים לחלקי מניות (share)

- Share.cs

מחלקה זו מכילה את כל הפרטים לרכישה של מניה של משתמש.

```
public required string Id { get; set; }
public required double PurchasingPrice { get; set; }
public required DateTime PurchaseDate { get; set; }
public required double Amount { get; set; }
```

- SharePurchase.cs

מחלקה זו מכילה את הפרטים הרלוונטיים של רכישה שבוצעה על ידי המשתמש לקנייה של מניה, לצורך יצירת share ושיוך למשתמש ותיק הרלוונטי.

```
public required string UserEmail { get; set; }
public required string StockSymbol { get; set; }
public required DateTime PurchaseDate { get; set; }
public required double PurchasingPrice { get; set; }
public required double Amount { get; set; }
public required string ListName { get; set; }
```

- ShareSale.cs

מחלקה זו מכילה פרטים רלוונטיים של מכירה של מניה של משתמש.

```
public required string UserEmail { get; set; }
public required string ListName { get; set; }
public required string StockSymbol { get; set; }
public required string SharePurchaseGuid { get; set; }
```

- StockListDetails.cs

מחלקה זו מכילה פרטים רלוונטיים לצורך שיוך למשתמש ותיק רלוונטי.

```
public required string UserEmail { get; set; }
public required string ListName { get; set; }
```

- WatchingStock.cs

מחלקה זו מכילה פרטים הרלוונטיים לצורך שמירת פרטי מניה בתוך תיק השקעות.

```
public required Dictionary<string, Share> PurchaseGuidToShares { get; set; }
public string? Note { get; set; }
```

- WatchingStockAction.cs

מחלקה זו מכילה פרטים רלוונטיים לצורך ביצוע הוספה או מחיקה של מניה מתיק השקעות.

```
public required string Email { get; set; }
public required string ListName { get; set; }
public required string StockSymbol { get; set; }
public string? Note { get; set; }
```

2.4.5. מודלים לחיפוש מניות (Search Result)

- SearchResult.cs

מודל חיפוש מניה, לצורך שמירה בבסיס הנתונים.

```
public required string Id { get; set; }
public required string SearchTerm { get; set; }
public required List<StockSearchResult> StockSearchResults { get; set; }
```

- StockSearchResult.cs

מודל תגובה לחיפוש מניה, מכיל את פרטים בסיסיים על מניה שנמצאה בחיפוש.

```
public required string Symbol { get; set; }
public required string Name { get; set; }
public required string ExchDisp { get; set; }
public required string TypeDisp { get; set; }
```

2.4.6. מודלים לטרנדים למניות (Market Trend)

- StockTrend.cs

מודל מניה הנמצאת בטרנד שבוצע עבורו חיפוש.

```
public string? Symbol { get; set; }
public string? Type { get; set; }
public string? Name { get; set; }
public double? Price { get; set; }
public double? Change { get; set; }
public double? ChangePercent { get; set; }
public double? PreviousClose { get; set; }
public double? PreOrPostMarket { get; set; }
public double? PreOrPostMarketChange { get; set; }
public double? PreOrPostMarketChangePercent { get; set; }
public string? LastUpdateUtc { get; set; }
public string? Currency { get; set; }
public string? Exchange { get; set; }
public string? ExchangeOpen { get; set; }
public string? ExchangeClose { get; set; }
public string? Timezone { get; set; }
public string? CountryCode { get; set; }
```

- MarketTrend.cs

מודל טרנד לפי שם הטרנד וערכי מניות בטרנד ותוצאות חיפוש.

```
public string? Id { get; set; }
public required string TrendName { get; set; }
public required List<StockTrend> TrendingStocks { get; set; }
public required List<StockNews> StockNews { get; set; }
public required DateTime LastUpdatedTime { get; set; }
```

- StockNews.cs

מודל תוצאת חדשות לטרנד.

```
public string? ArticleTitle { get; set; }
public string? ArticleUrl { get; set; }
public string? ArticlePhotoUrl { get; set; }
public string? Source { get; set; }
public string? PostTimeUtc { get; set; }
public List<StockTrend>? StocksInNews { get; set; }
```

2.4.7. ממשקים

- IAppConfiguration.cs

ממשק משותף לפרוייקט StocksApi, YahooFinance המאפשר קבלת מידע מקונפיגורציה בזמן ריצה.

```
T Get<T>(string section);
T Get<T>(params string[] sections);
```

2.5. YahooFinance

2.5.1. מודלים כלליים

- ApiConfiguration.cs

מודל קונפיגורציית api, מאפשר למשתמש לקבל את הקונפיגורציות של המערכת בזמן ריצה.

```
public required string BaseUrl { get; set; }
public required Dictionary<string, string> Headers { get; set; }
```

- PriceResponse.cs

מודל מחיר מנייה מתקבל מנתוני שוק ההון.

```
public required string Symbol { get; set; }
public required string ShortName { get; set; }
public required double RegularMarketPrice { get; set; }
public required double FiftyDayAverage { get; set; }
public required double TwoHundredDayAverage { get; set; }
public required string FiftyTwoWeekRange { get; set; }
public required double FiftyTwoWeekLow { get; set; }
public required double FiftyTwoWeekHigh { get; set; }
public string? AverageAnalystRating { get; set; }
```

2.5.2. מודלים YahooFinance15

- BasicResponse<T>.cs

מודל תגובה בסיסי לפי מודל T.

```
public required YahooMeta Meta { get; set; }
public required List<T> Body { get; set; }
```

- YahooMeta.cs

מודל מידע לתגובה, מודל לצורך deserialization בלבד.

```
public required string Version { get; set; }
public required int Status { get; set; }
public required string Copywrite { get; set; }
public required string Symbol { get; set; }
public required string ProcessedTime { get; set; }
```

2.5.3. מודלים YahooFinance127

- Finance127AnalysisResponse

מודל תחזית ערך מניה עתידי.

```
public required RawFmtValue CurrentPrice { get; set; }
public required RawFmtValue TargetHighPrice { get; set; }
public required RawFmtValue TargetLowPrice { get; set; }
public required RawFmtValue TargetMeanPrice { get; set; }
public required RawFmtValue TargetMedianPrice { get; set; }
```

- RawFmtValue.cs

מודל עזר לקבלת ערך מניה.

```
public double Raw { get; set; }
public required string Fmt { get; set; }
```

2.5.4. מודלים YahooFinance1

Finance1BulkResponse.cs

מודל תגובה של מספר ערכי מניות

```
public required StocksResult QuoteResponse { get; set; }
```

- StocksResult.cs

מודל תגובה של מספר מניות

```
public required List<PriceResponse> Result { get; set; }
```

- Finance1SearchResponse.cs

מודל חיפוש מניות לפי שם

```
public required List<Quote> Quotes { get; set; }
public required List<StockNews> News { get; set; }
```

- StockNews.cs

מודל חדשות על מניות.

```
public required string Title { get; set; }
public required string Publisher { get; set; }
public required string Link { get; set; }
public required string Type { get; set; }
public required List<string> RelatedTickers { get; set; }
```

- Quote.cs

מודל תגובה למניות בחיפוש טרנדים.

```
public required string Symbol { get; set; }
public required string ShortName { get; set; }
public required string Exchange { get; set; }
public required string ExchDisp { get; set; }
public required string TypeDisp { get; set; }
```

2.5.5. מודלים RealTimeFinanceData

- MarketNewsResponse.cs

מודל לקבלת חדשות על טרנד

```
public string? Article_Title { get; set; }
public string? Article_Url { get; set; }
public string? Article_Photo_Url { get; set; }
public string? Source { get; set; }
public string? Post_Time_Utc { get; set; }
public List<MarketTrendResponse>? Stocks_In_News { get; set; }
```

- MarketTrendsResponse.cs

מודל קבלת תגובת טרנד.

```
public required List<MarketTrendResponse> Trends { get; set; }
public required List<MarketNewsResponse> News { get; set; }
```

- MarketTrendResponse.cs

מודל קבלת פרטי מניה בטרנד

```
public string? Symbol { get; set; }
public string? Type { get; set; }
public string? Name { get; set; }
public double? Price { get; set; }
public double? Change { get; set; }
public double? Change_Percent { get; set; }
public double? Previous_Close { get; set; }
public double? Pre_Or_Post_Market { get; set; }
public double? Pre_Or_Post_Market_Change { get; set; }
public double? Pre_Or_Post_Market_Change_Percent { get; set; }
public string? Last_Update_Utc { get; set; }
public string? Currency { get; set; }
public string? Exchange { get; set; }
public string? Exchange_Open { get; set; }
public string? Exchange_Close { get; set; }
public string? Timezone { get; set; }
public string? Country_Code { get; set; }
```

- RealTimeFinanceBasicResponse.cs

מודל בסיסי של קבלת מידע בRealTimeFinance לצורך דסיריאליזציה.

```
public required string Status { get; set; }
public required string Request_Id { get; set; }
public required MarketTrendsResponse Data { get; set; }
```

2.5.6. ממשקים

- IFinanceStrategy.cs

ממשק לקבלת נתוני שוק ההון בזמן אמת באמצעות מספר שרתים שונים בצורה מנוהלת.

```
Task<List<StockSearchResult>> FindStockAsync(string searchTerm);
Task<PriceResponse?> GetStockAsync(string symbol);
Task<List<StockAnalysis>> GetStocksAnalysisAsync(string[] symbols);
Task<List<PriceResponse>> GetStocksAsync(params string[] symbols);
```

- IRealTimeFinanceData.cs

ממשק לקבלת טרנדים למניות לפי שם.

```
Task<MarketTrendsResponse?> GetMarketTrendAsync(string trendType);
```

- IStockAnalysisApi.cs

ממשק לקבלת ניתוחים עתידיים לערכי מניות.

```
Task<StockAnalysis?> GetStockAnalysisAsync(string symbol);
Task<List<StockAnalysis>> GetStocksAnalysisAsync(params string[] symbols);
```

- IWebApi.cs

ממשק פניה לשרתים חיצוניים לקבלת נתונים שונים באמצעות בקשות Http.

```
Task<T?> GetResponseAsync<T>(string endpoint,
        params KeyValuePair<string, string>[] queryParams);
Task<RestResponse> GetResponseAsync(string endPoint,
        params KeyValuePair<string, string>[] queryParams);
```

- IYahooFinance.cs

ממשק לקבלת ערכי מניה בזמן אמת משרת חיצוני.

```
Task<PriceResponse?> GetStockAsync(string symbol);
Task<List<StockSearchResult>> FindStockAsync(string searchTerm);
```

- IYahooFinanceBulk.cs

ממשק לקבלת ערכים של מספר מניות בבקשה אחת בזמן אמת משרת חיצוני.

```
Task<List<PriceResponse>> GetStocksAsync(params string[] symbols);
```

2.5.7. מחלקות

- StockSearchInformationGenerator.cs

מחלקה סטטית ליצירת StockSearchResult מתגובה Quote של YahooFinance1.

```
public static StockSearchResult Generate(Quote quote);
```

- StockAnalysisGenerator.cs

מחלקה סטטית ליצירת StockAnalysis במקרה שלנו קיימת שיטה אחת המקבלת Finance127AnalysisResponse של YahooFinance127 וממירה אותו לStockAnalysis.

```
public static StockAnalysis Generate(string symbol,
        Finance127AnalysisResponse response)
```

- WebApiFactory.cs

מפעל לייצור IWebApi שונים באמצעות קובץ הקונפיגורציה.
משתמש במימוש של הממשק IAppConfiguration.

```
public IWebApi Generate(string section);
```

- QueueExtensions.cs

מחלקה סטטית בהוספת פונקציונליות נדרשת למחלקה Queue.
הוספה של שיטה המאפשרת עבודה עם Queue של api חיצוניים והפעלה של פקודה מכל api לפי הסדר.
באמצעות תוספת זו מקבלים מידע על שוק ההון מיותר מספק אחד בתור אחד אחרי השני (בעקבות עבודה עם api חיצוניים חינמיים עם מגבלות יומיות).

```
public static async Task<TResult> ExecuteWithQueueAsync<T,
        TResult>(this Queue<T> queue, Func<T, Task<TResult>> action)
```

- YahooFinanceApis

מחלקות המאפשרות קבלת נתוני שוק ההון, קיימות 4 מחלקות שונות כך שכולן משתמשות במימוש למחלקה WebApiFactory לצורך יצירת IWebApi חדש לבקשות Http.
המחלקות: RealTimeFinanceData, YahooFinance1, YahooFinance15, YahooFinance127.
כל מחלקה מממשת את הממשק הייעודי לה.

- FinanceStrategy.cs

מחלקה המממשת את הממשק IFinanceStrategy.
המחלקה מפעילה 3 Queue שונים לצורך ביצוע בקשות Http השונות לשרתים לקבלת מידע משוק ההון.
המחלקה משתמשת גם במימוש של QueueExtensions כדי לאפשר מעבר בין Api שונים.
בנוסף המחלקה מחזיקה את כל המימושים שקיימים בפרוייקט הריצה של IYahooFinance, IStockAnalysisApi כרשימות המומרות לQueue.

- WebApi.cs

מחלקה המממשת את ממשק IWebApi.
מחלקה זו מעבירה בקשות Http לשרתים חיצוניים לצורך קבלת נתונים, במקרה שלנו קבלת נתוני שוק ההון, חיפוש מניות וטרנדים.
המחלקה משתמשת במחלקה חיצונית RestClient לצורך העברת בקשות Http.

3. ממשק משתמש

3.1. כללי

ממשק המשתמש מאפשר אינטרקציה של המשתמש עם המערכת.
המערכת מותאמת למחשבים נייחים בלבד ואינו מותאם לפלאפונים.
המערכת נוחה לשימוש ולניווט בין המסכים השונים.
חלונות צפים קיימים במקומות בהם נדרש מהמשתמש להכניס קלט לצורך ביצוע פעולות או לשם אישור פעולה בלתי הפיכה.
סרגל הניווט משתנה לאחר ההתחברות ומאפשר פעולות נוספות למשתמשים רשומים בלבד.

3.2. עקרונות מנחים

- נוח לשימוש, קל לניווט
- מספק מידע עדכני ככל שניתן לערכי מניות
- טבלאות רוויות מידע מאפשרות ניווט בין דפים ומיון הנתונים.
- שימוש בדפוסי סגנון הקיימים במערכת הבסיס והוספת ממשקים נוספים החשובים לצורך יעילות המערכת.
- פתוח לשינויים ושיפורים עתידיים.

3.3. מודלים

כלל המודלים של ממשק המשתמש הם מודלים הקיימים בשרת ולכן אין מודלים נוספים.

3.4. ממשקים

3.4.1. Observable

ממשק מובנה מחבילה RxJS המאפשר עבודה עם נתונים א-סינכרוניים בצורה יעילה באמצעות מנגנון של "הרשמה" (subscription).
הפעולה הראשונה בה נעשה שימוש היא subscribe, שמאפשרת לנו להשתמש בערך שמתקבל מהפעולה.
פעולה נוספת היא pipe, שמאפשרת לבצע שרשור של פעולות שונות על הנתונים לפני החזרתם, לדוגמה: שינוי הערכים, טיפול בשגיאות, סינון הנתונים ועוד.
חשוב לציין שהפעולות ב-Observable משורשרות זו לזו, ולכן יש למקם את pipe לפני subscribe.
השימוש ב-Observable במערכת שלנו מתבצע בעיקר בעת ביצוע בקשות HTTP. אנו משתמשים ב-pipe לצורך הצגת מסך טעינה, לטיפול בשגיאות המתקבלות מהשרת, ולפעמים גם לשינוי הערכים.
כדי לקבל את הנתונים ולפעול עליהם, יש לבצע subscribe, שהיא הפעולה הבסיסית הנדרשת כדי להפעיל את ה-Observable ולגשת לערכים שהתקבלו.

3.4.4. Route Guards

ממשק מובנה ב-Angular שמאפשר לבדוק נתונים או תנאים מסוימים לפני גישה לנתיב (Route), כך ניתן לבצע מידור מידע בהתאם להרשאות המשתמש.
ממשק זה מאפשר שכבת אבטחה נוספת המונעת גישה לנתיבים רגישים ללא הרשאות מתאימות.
ניתן לעשות שימוש בCanActivate, CanActivateChild אשר מטרתם היא לבדוק האם ניתן לגשת לנתיב והאם ניתן לגשת לבניו של אחד מהנתיבים.

3.4.5. Interceptor

ממשק מובנה ב-Angular שמאפשר לבצע פעולות על בקשות HTTP באופן גלובלי, לפני שליחתן או לאחר קבלת התשובה.
באמצעות ממשק זה נוכל גם לשנות תגובות של הבקשות ולהעביר שגיאות עבור נתונים מסוימים.

3.5. מחלקות

3.5.1. Factories

- MessageFactory.ts

מחלקה ליצירת הודעות pop-up למשתמש בעת שגיאה, הצלחה ומידע.

```
public createErrorMessage(body?: string): Message;
public createSuccessMessage(body?: string): Message;
public createInfoMessage(body?: string): Message;
```

3.5.2. Services

מחלקות אשר מנגישות לנו פונקציונליות רלוונטית לאתר.

- CookieService.ts

מחלקה לשימוש בקוקיז של הדפדפן.

```
getCookie(name: string): string | null;
setCookie(name: string, value: string, days?: number): void;
deleteCookie(name: string): void;
```

- AuthenticationService.ts

מחלקה לביצוע בדיקות חיבור של המשתמש

```
getUserEmail(): string | null;
disconnectUser();
userConnection(): Observable<boolean>;
isUserConnected(): boolean;
updateConnectedUser(email: string): void;
```

- LoadingService.ts

מחלקה להצגה והסתרה של מסך טעינה בעת ביצוע בקשות http

```
loading$: Observable<boolean>;
show(): void;
hide(): void;
```

- ToastService.ts

מחלקה להצגת טוסטים (התראות במסך), מחלקה זו עושה שימוש בMessageFactory כדי ליצור את ההודעה הנכונה למשתמש.
מחלקה זו עושה שימוש במחלקה MessageService שהיא מחלקה חיצונית (PrimeNg) המאפשרת הופעת טוסטים.

```
addSuccessMessage(body: string): void;
addErrorMessage(body: string): void;
addInfoMessage(body: string): void;
```

- SharesService.ts

מחלקה לביצוע בקשות http מול שרת בהקשר של חלקי מניות.
מבצעת שימוש במחלקה HttpClient לצורך ביצוע הבקשות.

```
addUserShare(sharePurchase: SharePurchase): Observable<Share | null>;
removeUserShare(shareSale: ShareSale): Observable<boolean>;
addWatchingStock(email: string, listName: string, stockSymbol: string):
        Observable<boolean>;
removeWatchingStock(email: string, listName: string, stockSymbol: string):
        Observable<boolean>;
updateWatchingStockNote(email: string, listName: string, stockSymbol: string,
        note: string): Observable<boolean>;
addUserList(stockListDetails: StockListDetails): Observable<boolean>;
removeUserList(stockListDetails: StockListDetails): Observable<boolean>;
```

- StockService.ts

מחלקה לביצוע בקשות http מול שרת בהקשר של מניות

```
getAllStocks(): Observable<Stock[]>;
getStockBySymbol(stockSymbol: string): Observable<Stock>;
findStocksBySearchTerm(searchTerm: string): Observable<Stock[]>;
getMarketsTrends(): Observable<MarketTrend[]>;
shouldBeUpdated(startTime: Date | undefined, endTime: Date): boolean;
```

- UserService.ts

מחלקה לביצוע בקשות http מול שרת בהקשר של משתמשים

```
getUser(): Observable<User>;
createUser(user: UserCreation): Observable<boolean>;
addStockNotification(stockNotification: StockNotification):
        Observable<ObjectIdResponse>;
removeStockNotification(email: string, notificationId: string): Observable<boolean>;
tryConnect(email: string, password: string): Observable<boolean>;
updatePassword(email: string, password: string): Observable<boolean>;
```

3.5.3. RouteGuard

- connectedUserGuard.ts

מחלקה המממשת ממשק RouteGuard ומבצעת בדיקה האם משתמש מחובר בעת ניסיון כניסה לדפים שאינם מורשים לאורחים.
מחלקה זו עושה שימוש במימוש של המחלקה Router כדי לנווט את המשתמש לדף ההתחברות ועושה שימוש במימוש של המחלקה AuthenticationService כדי לבדוק האם המשתמש מחובר.

3.5.4. Interceptor

- connectionInterceptor.ts

מחלקה המממשת ממשק Interceptor ומבצעת בדיקה כי בקשות Http יוצאות אינן אורכות יותר מזמן מסויים שנקבע, זאת על מנת לאפשר בדיקה האם המשתמש מחובר לשרת או אם יש בעיית שרת במידה וישנה בעיה תינתן הודעת שגיאה.

- loadingInterceptor.ts

מחלקה המממשת את ממשק Interceptor ומציגה או מסתירה את מסך הטעינה בעת שליחת בקשות Http וכך מציגה למשתמש כי עליו להמתין בעת ביצוע בקשה.

3.6. מסכים

הממשק הגרפי מבוסס על עקרון הפרדה לחלקים (Components), כל רכיב אחראי על פונקציה או חלק מסוים בממשק המשתמש.

המסך הראשי של האפליקציה מיוצג על ידי רכיב app-root, שהוא הרכיב הראשי של המערכת, וכולל את כל הרכיבים האחרים שמרכיבים את ממשק המשתמש.

ההיררכיה של הרכיבים בתוך app-root היא כדלקמן:

```
app-root
├── app-header
├── router-outlet
│  ├── app-home
│  ├── app-login
│  ├── app-register
│  ├── app-search
│  ├── app-stock-details
│  ├── app-stocks-table
│  ├── app-insights
│  ├── app-user-stocks
│  ├── app-stock-shares
│  ├── app-user-information
│  ├── app-notification-center
│  └── app-not-found
├── app-footer
├── app-loading
├── p-confirmDialog
└── p-toast
```

3.6.1. רכיבים חיצוניים

רכיבים מחבילות חיצוניות המספקות לנו פונקציונליות כללית שניתן להשתמש בכלל האתר.

p-toast
רכיב זה מבוסס על חבילה חיצונית ומאפשר קבלה של טוסטים (הודעות קופצות) למסך המשתמש בעת ביצוע פעולות תקינות או שגיאות.

p-confirmDialog
רכיב חיצוני נוסף מאפשר הצגה של חלון בחירה למשתמש כדי שיוכל לאשר פעולות או לבצע פעולות הדורשות קלט.

p-table
רכיב מבוסס על חבילה חיצונית שמאפשר שימוש בנתונים כרשימה לצורך יצירת טבלה חכמה עם יכולות כמו מיון, דפים וכו'.
רכיב זה בשימוש במספר רכיבים לדוגמה במסך הצגת כלל המניות.

3.6.2. רכיבי התצוגה לכלל המשתמשים

app-loading
מסך טעינה המוצג בכל פעם שנוצרת בקשה לשרת ומציגה עיגול טעינה המסתיימת בעת סיום קבלת תגובה מהשרת.

app-header
רכיב ניווט - מציג את כלל האופציות לניווט באתר, בנוסף כאשר המשתמש מחובר ניתן לבצע פעולות נוספות המאופיינות רק למשתמש מחובר (לדוגמה מעבר לניהול תיקי מניות, התראות וכו').

router-outlet
רכיב דינמי מאפשר הצגה של רכיבים שונים לפי הנתיב הנוכחי באתר, כך ניתן להציג מסכים שונים באתר.

app-footer
רכיב המציג שורה בתחתית הדף של trade mark.

app-home
מציגה את המסך הראשי של האפליקציה, המסך שהמשתמש רואה לראשונה ונותן אופציה לגשת לאיזור ההתחברות.

app-login
איזור ההתחברות - מאפשר למשתמש להתחבר, לשחזר סיסמה ובנוסף נותן הפניה למסך ההרשמה.

app-register
מסך ההרשמה - מאפשר למשתמש שאינו רשום לבצע רישום לאתר באמצעות פרטים אישיים וסיסמה.

app-search
מסך חיפוש מניות - מאפשר למשתמש מחובר ולא מחובר לחפש מניות ולאחר מכן נותן קישור למסך צפיה במניה לכל אחת מהמניות שמצא.

app-stock-details

מסך צפיה במניה - מציג פרטי מניה מורחבים, ניתוח בסיס על המניה לצורך מתן אופציה למשתמש לנתח את ערך המניה העתידי, במסך זה ניתנת אופציה להוסיף התראות לערך מניה עתידי והוספת מניה לתיק במידה והמשתמש מחובר.

app-stocks-table

מסך להצגת כלל המניות הרשומות באתר - מציג את כל המניות הרשומות באתר (אדם נכנס למסך הצפיה שלהם) ואחריהן האתר ממשיך לעקוב באופן אוטומטי. מציגה את המניות בטבלה עם פרטים בסיסיים ואפשרות מעבר למסך הצפייה במניה.

app-not-found

מסך שגיאה כאשר משתמש מגיע לנתיב שאינו קיים.

3.6.2. רכיבי התצוגה למשתמשים מחוברים בלבד

app-insights
מסך לצפייה בטרנדים - מציג את כל הטרנדים, מניות הקשורות לטרנד הספציפי ובנוסף חדשות הקשורות לטרנד. מסך זה מכיל קישורים לכל המניות וקישור חיצוני לצפייה בחדשות.

app-user-stocks
מסך לצפייה בתיקי מניות - מציג את כל התיקים שקיימים אצל המשתמש, בנוסף לאחר בחירה ניתן לצפות בכל תיק ולבצע פעולות הקשורות לתיק (הוספת מניות, הוספת חלקי מניות וכו').

app-portfolio-detail
מסך לצפייה בכל תיק מניות בנפרד, למעשה מסך הצפייה בתיקים מכיל את הקומפוננטה הזאת באמצעות נתון המשתנה והוא התיק הנבחר.

app-user-information
מסך לצפייה בפרטי משתמש - מציג פרטים כללים על כל התיקים בהם יש חלקי מניות, מציג פרטים על המשתמש ורווחים / הפסדים.

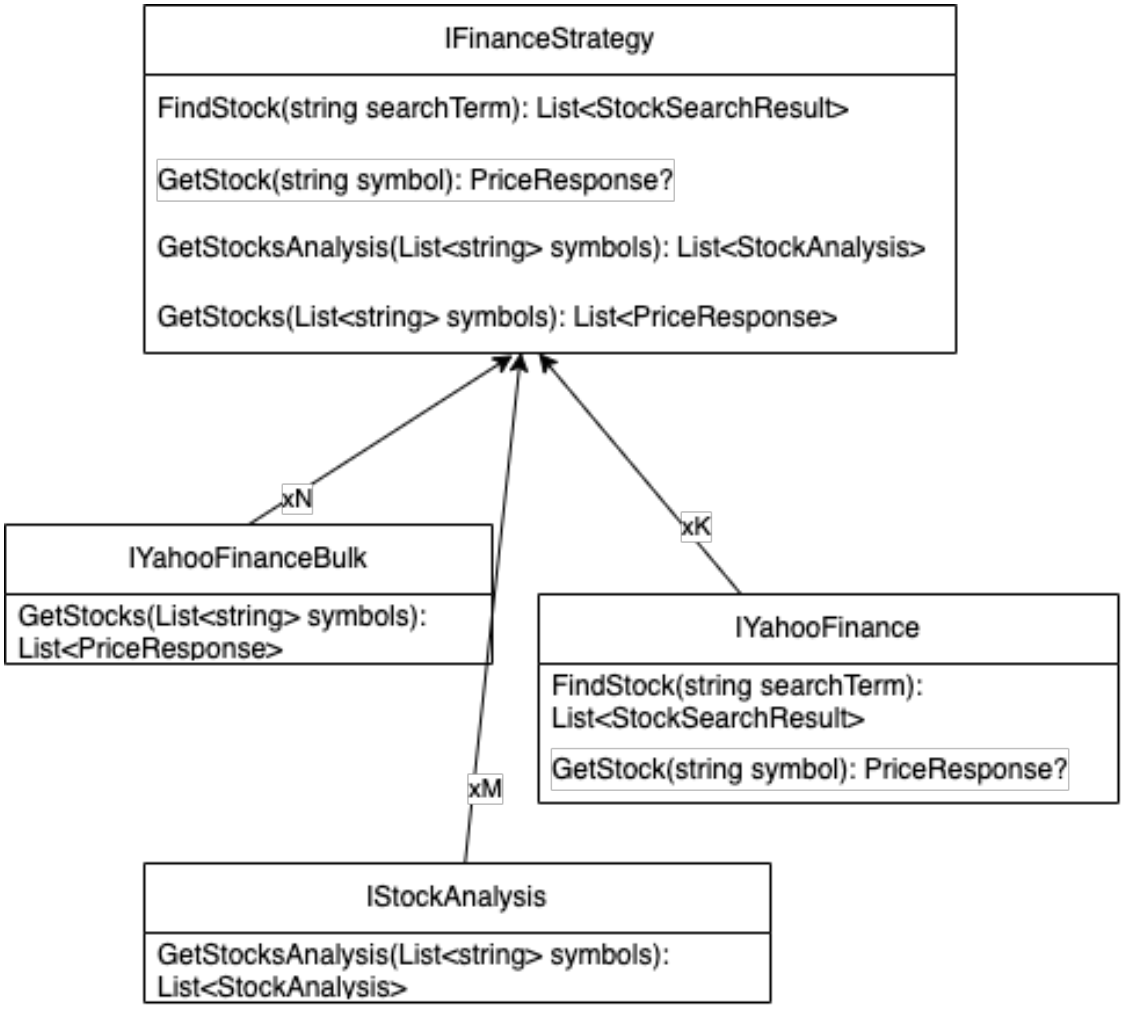
app-notification-center
מסך לצפייה בהתראות מחירים של המשתמש, המשתמש יכול לצפות בכל ההתראות ולראות איזה מהן פעילה (ההתראה הופעלה על ידי האתר) וכמו כן למחוק התראות.

app-stock-shares
מסך שמאפשר הצגה של חלקי המניות שיש למשתמש ממניה מסוימת בתיק, במסך זה ניתן להוסיף חלקי מניות נוספים (מדמה רכישה של מניות) לפי כמות, מחיר ממוצע ותאריך הרכישה.

4. דפוסי עיצוב

4.1. Strategy

אנחנו משתמשים בדפוס סגנון strategy על מנת שנוכל להוסיף api נוספים לצורך עדכון מניות וחיפוש.



4.2. Dependency injection

כלל השרת מבוסס על Dependency Injection המאפשר החלפה דינמית של שירותים (Services) באמצעות מימושים שונים של ממשקים משותפים. מערכות כמו .NET ו-Angular תומכות במנגנון Dependency Injection ברמת הבסיס, ומאפשרות מבנה גמיש וקל לניהול.

בDotNET, ניתן להגדיר את השירותים (Services) בקובץ Startup.cs או בקובץ Program.cs.

הגדרה זו נעשית באמצעות ממשק ה-IServiceCollection המספק שיטות להוספת שירותים, ואופן ניהולם (Singleton, Scoped, או Transient). לאחר ההגדרה, כל פעם שמחלקה זקוקה לשירות מסוים, ניתן לקבל Instance מתאים דרך הקונסטרקטור, והמערכת תדאג לספק את השירות באופן אוטומטי.

מנגנון ה-Dependency Injection של DotNET מאפשר לנהל את השירותים בצורה ריכוזית וגמישה, ומקל על התאמה ושינוי של התנהגות המערכת בעתיד. השימוש ב-IServiceCollection לא רק מפשט את תהליך הפיתוח, אלא גם תורם לכתיבת קוד נקי יותר, בעל תלותיות נמוכה וקלה יותר לבדיקה.

דוגמה להגדרת Singleton:

```
services.AddSingleton<IStocksDal, StocksDal>();
```

בדומה ניתן לבצע הוספה לservices כScoped, Transient באמצעות שינוי המילה Singleton.

דוגמה לקבלת המחלקה בקונסטרקטור:

```
public StockRepository(
        IStockDal stocksDal,
        ILogger<Stock> logger,
        IFinanceStrategy yahooFinanceStrategy)
{
        _stocksDal = stocksDal;
        _logger = logger;
        _yahooFinanceStrategy = yahooFinanceStrategy;
}
```

4.3. Singleton

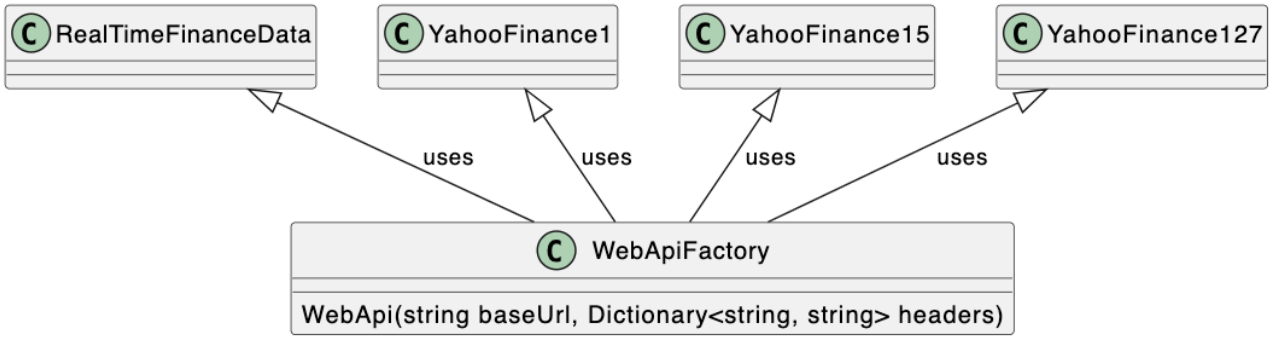
כלל המחלקות בשרת המבוססות על dependency injection אנחנו מוסיפים באמצעות singleton כדי לקבל מחלקה אחת המשותפת לכל המחלקות השונות בהם אנחנו משתמשים.
מתבצע כפי שתואר בסעיף 4.2, במקרה הזה יש להוסיף את המחלקה כSingleton ולכן יש להשתמש בפונקציה AddSingleton.

4.4. Transient

דפוס הסגנון transient בשימוש במחלקות של YahooFinance כך שבכל פעם שנרצה ליצור IWebApi נשתמש בfactory אשר יוצר לנו מחלקה חדשה לכל פעם שמתבקש ממנו ליצור webApi נוסף לשירות אחר ממנו אנחנו מביאים את נתוני המניות השונים.
בצורה הזו יתקבל כי לכל מחלקה שנדרשת להביא נתונים משרתים שונים יהיה WebApi משלו ולכן לא יהיה שימוש ממספר מקומות שונים.

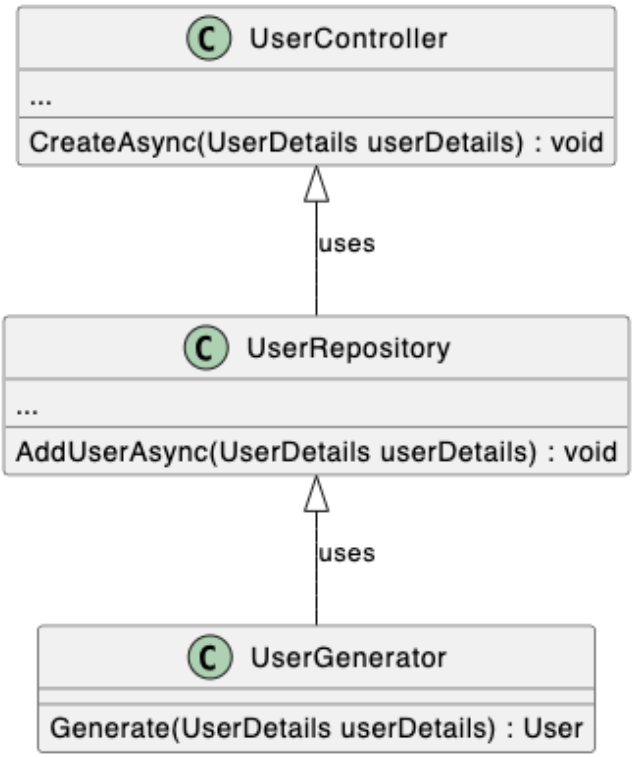
4.5. Factory

על מנת שנוכל לאפשר עבודה במקביל של מספר webapi השונים שנדרשים לעבוד במקביל נשתמש במפעל המייצר חיבורים מנותקים אחד מהשני לצורך קבלת נתונים משירותים חיצוניים המספקים ערכי מניות, חיפוש מניות וטרנדים.
המחלקה מתוארת בסעיף 2.5.7.



4.6. Generator

אנחנו משתמשים בדפוס סגנון generator כדי לשנות ערכים המתקבלים על ידי המשתמש והשלמת שדות חסרים למודל במסד הנתונים שלנו.



5. שינויים עתידיים

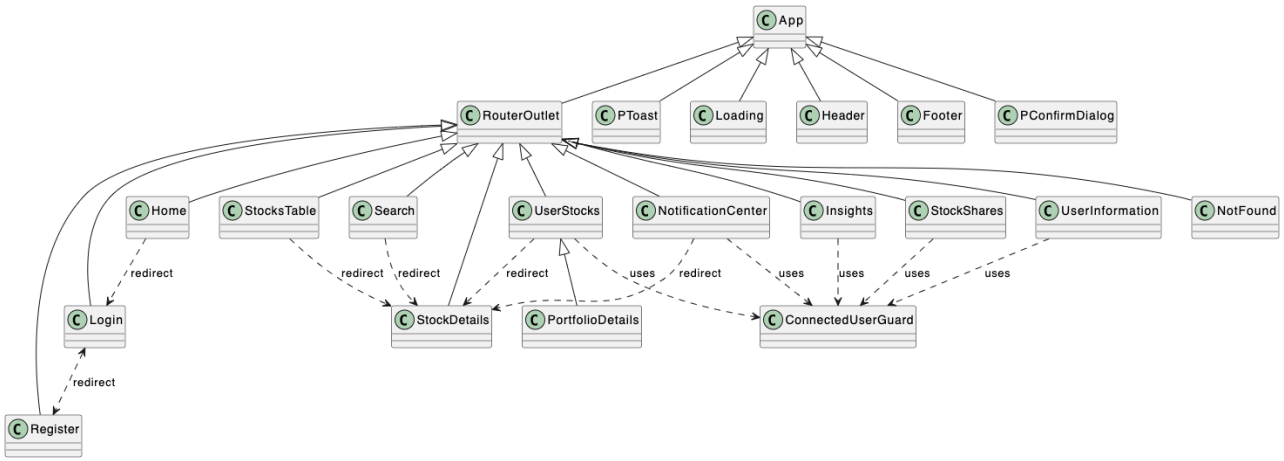
1. ניתן להוסיף אופציה לקבלת מידע על משתמש וביצוע פעולות על משתמש ללא שליחה של אימייל בבקשת http, כמו שקורה כעת.
    1. ניתן להוסיף בצד השרת מחלקה session-id, הtoken יוחזר בעת ניסיון התחברות והשרת ידע לשייך משתמש לכל token כולל זמן התפוגה.
    2. בעת קבלת בקשות http השרת ידרוש במקרים מסויימים לקבל מהמשתמש session-id בעת שליחת הבקשה כheader.
    3. ממשק המשתמש יצטרך לממש מחלקה שתפקידה יהיה ניהול session-id באמצעות שמירה כקוקיז, ובכל בקשת Http יצרף את הsession-id.
2. ניתן להוסיף בעתיד תמיכה בעוד סוגי מניות משווקים שונים.
    1. נדרש עדכון המודל של המניה כך שלכל מניה יהיה שיוך לשוק שונה (לדוגמה nyse, nasdaq, tel aviv וכו').
    2. המערכת תדע לפי המניה מה מאגר הנתונים שאליו היא אמורה לפנות לקבלת המידע העדכני לערך המניה.
    3. ממשק המשתמש ידע להציג את שם השוק בעת הצגת המניה.
3. הוספת מחיקת משתמש דרך הממשק הגרפי.
    1. יש לממש אופציה של מחיקה כendpoint לusersController.
    2. יש להוסיף אופציה למחיקה של משתמש באמצעות usersRepository.
    3. יש להוסיף אופציה של מחיקת משתמש כחלק ממסך פרטי המשתמש.
4. הוספת גרפים למניות, ניתוח גרפים והשוואת מניות בגרף אחד, בין היתר נדרש להוסיף ערכי מניות מהעבר.
    1. יש להוסיף למודל המניה שמירה של ערכי מניה לכל יום כרשימה ממודל חדש שמכיר את ערך המניה ואת התאריך.
    2. יש להוסיף בממשק המשתמש ניתוח של רשימת ערכי המניות ובכך נוכל ליצור גרפים ולבצע ניתוחים.
5. הוספת אופציה לרכישת מניות אקטיבית או חיבור ברוקרים קיימים לפלטפורמה שלנו.
    1. יש להוסיף בצד השרת חיבור מסוג oAuth לברוקרים השונים אותם המשתמש רוצה לחבר אליו.
    2. ממשק המשתמש ינגיש את הברוקרים אשר עובדים עם האתר ויציג מסך המאפשר חיבור אל הברוקר הרלוונטי.

6. הוספת שמירה של זיכרון בcache לביצוע פעולות מהיר יותר בעת בקשה לערכי מניה אשר תדירות הבקשות שלהם גבוהה יותר.
   1. יש להוסיף מחלקה בצד השרת השומרת את ערכי המניה ולפי הביקוש יודעת האם להמשיך לשמור כזיכרון מהיר ולא לבקש בקשה מבסיס הנתונים.
   2. ערכי המניה שישמרו בזיכרון מהיר זה יצטרכו להיות מעודכנים לפי תדירות העדכון שהוגדרה כדי שיהיו תמיד נאמנים לתוצאות שקיימות בבסיס הנתונים.
   3. בעת ביצוע העדכון היומי המערכת תדע לעדכן גם ערכי מניות קיימים בזיכרון המהיר.
7. הוספת פרטים נוספים לכל מניה, מאתרים שונים.
   1. ניתן לעדכן את מודל המניה ובכך להוסיף פרטים נוספים הרלוונטיים לגביהם.
   2. נדרש שממשק המשתמש יעדכן את הפרטים שהוא מציג.
8. הוספת סיכום רווחים לכל תיק בנפרד.
   1. יש לעדכן את מסך הצגת תיק כך שיוכל להציג סיכום לכל תיק בנפרד.

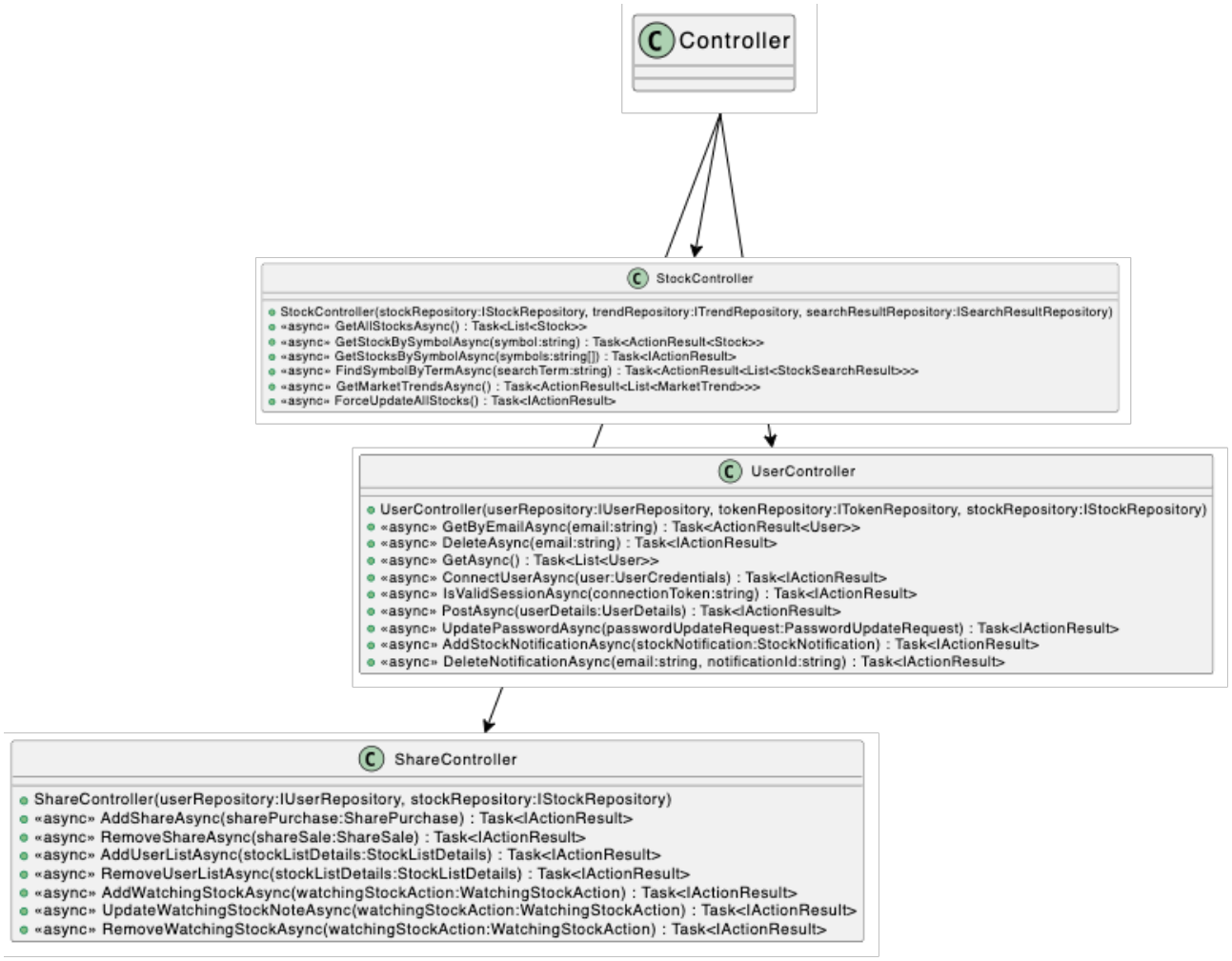
6. דיאגרמות

6.1. Class Diagrams

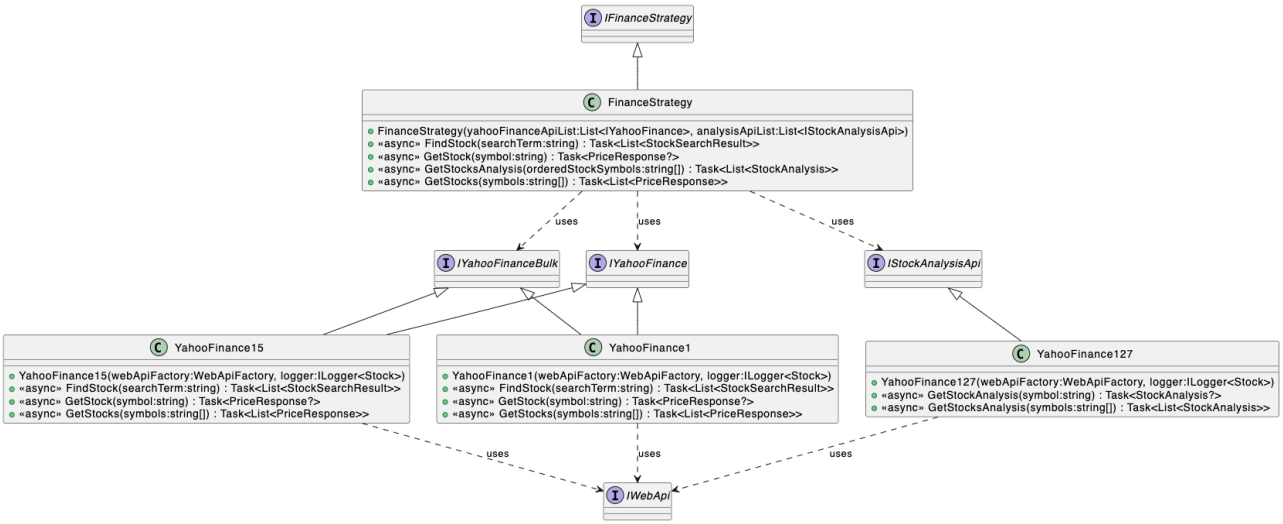
6.1.1. מסכים ממשק משתמש



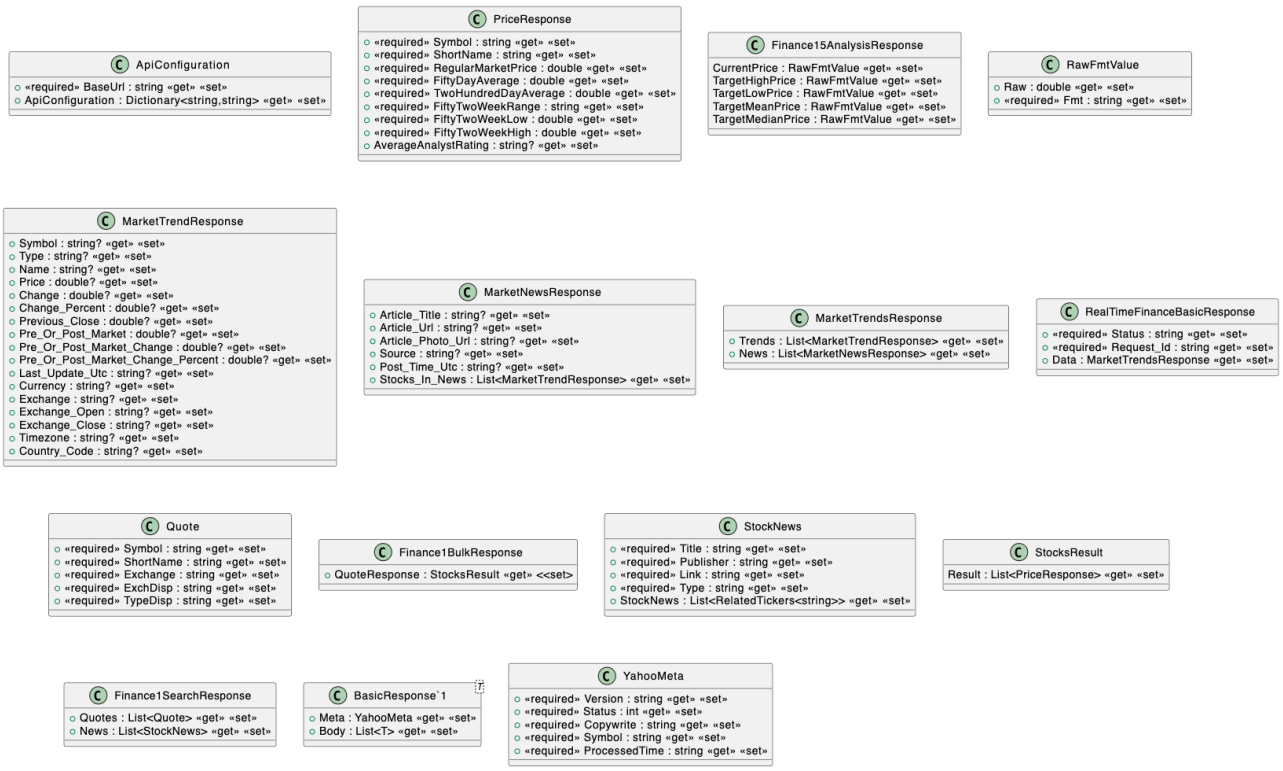
6.1.2. ממשק השרת



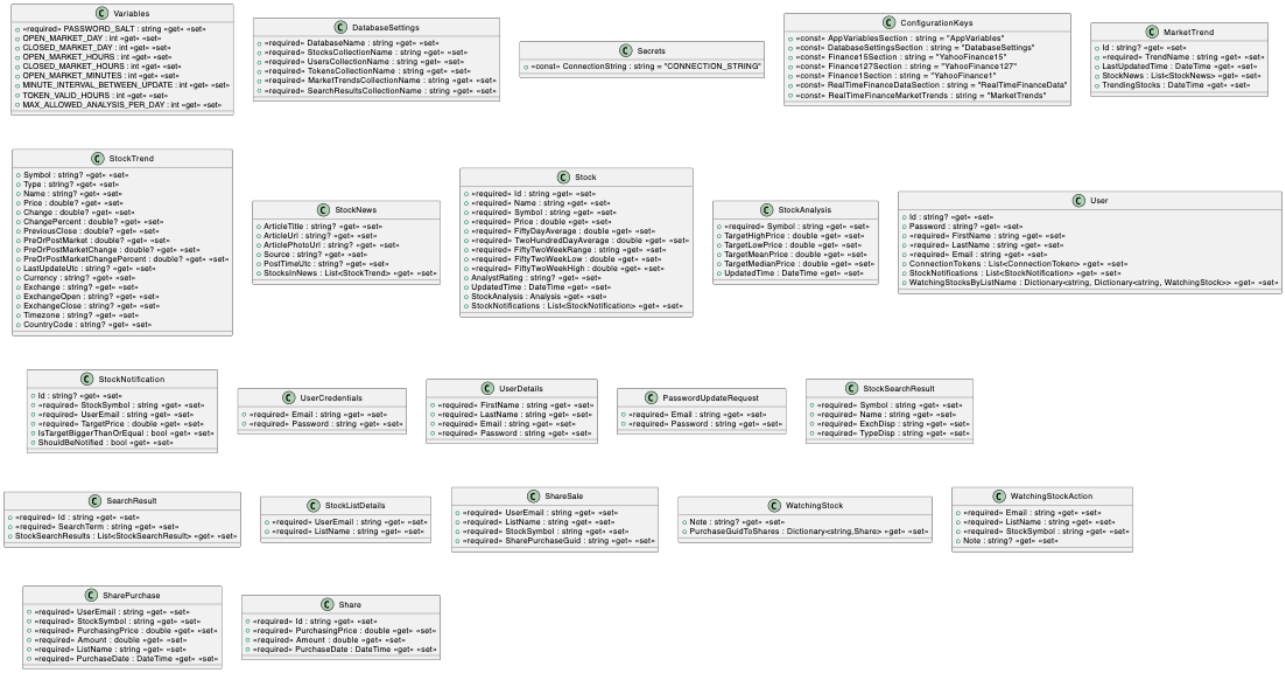
Finance Strategy .6.1.3



Yahoo Finance Models .6.1.4

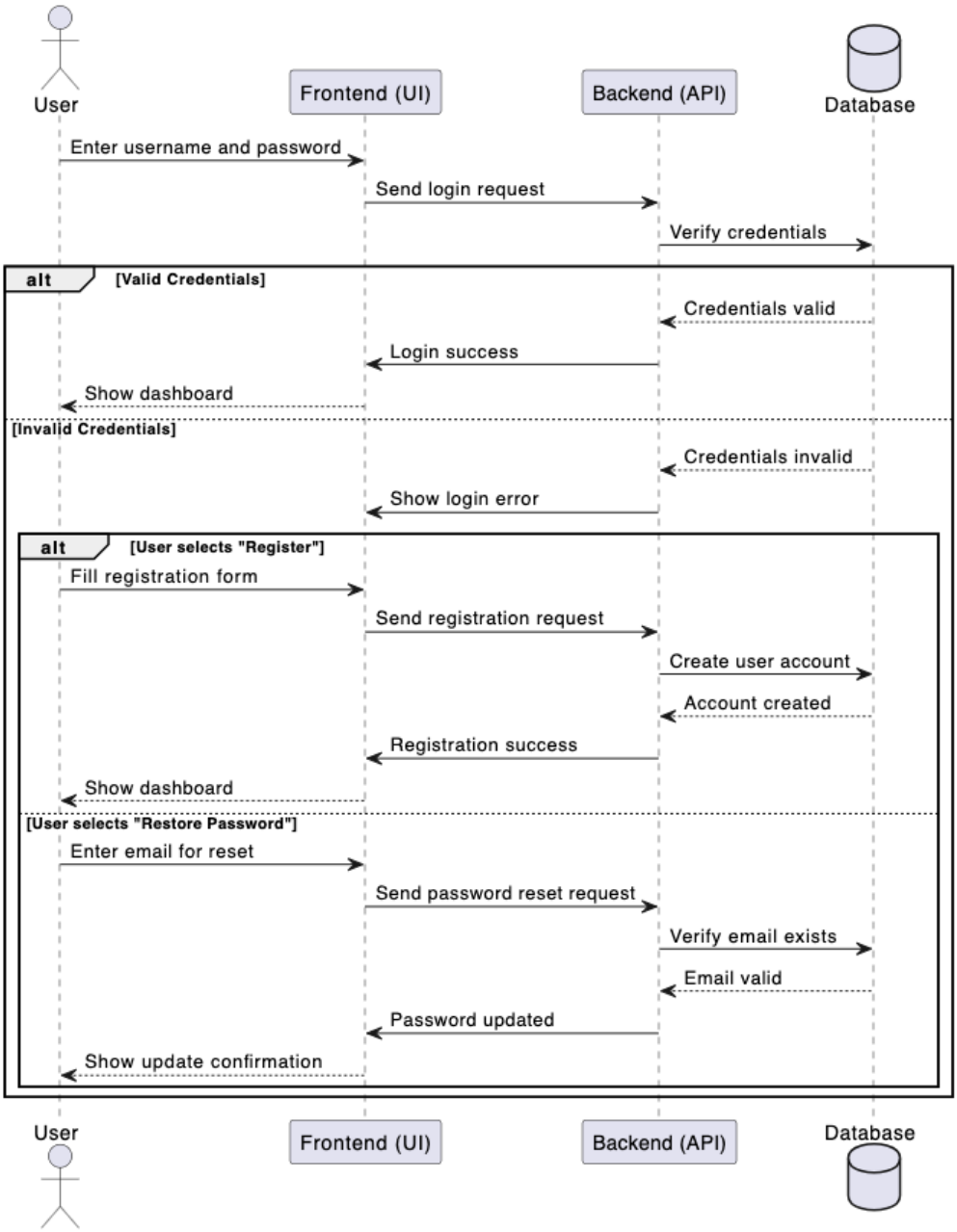


Library .6.1.5

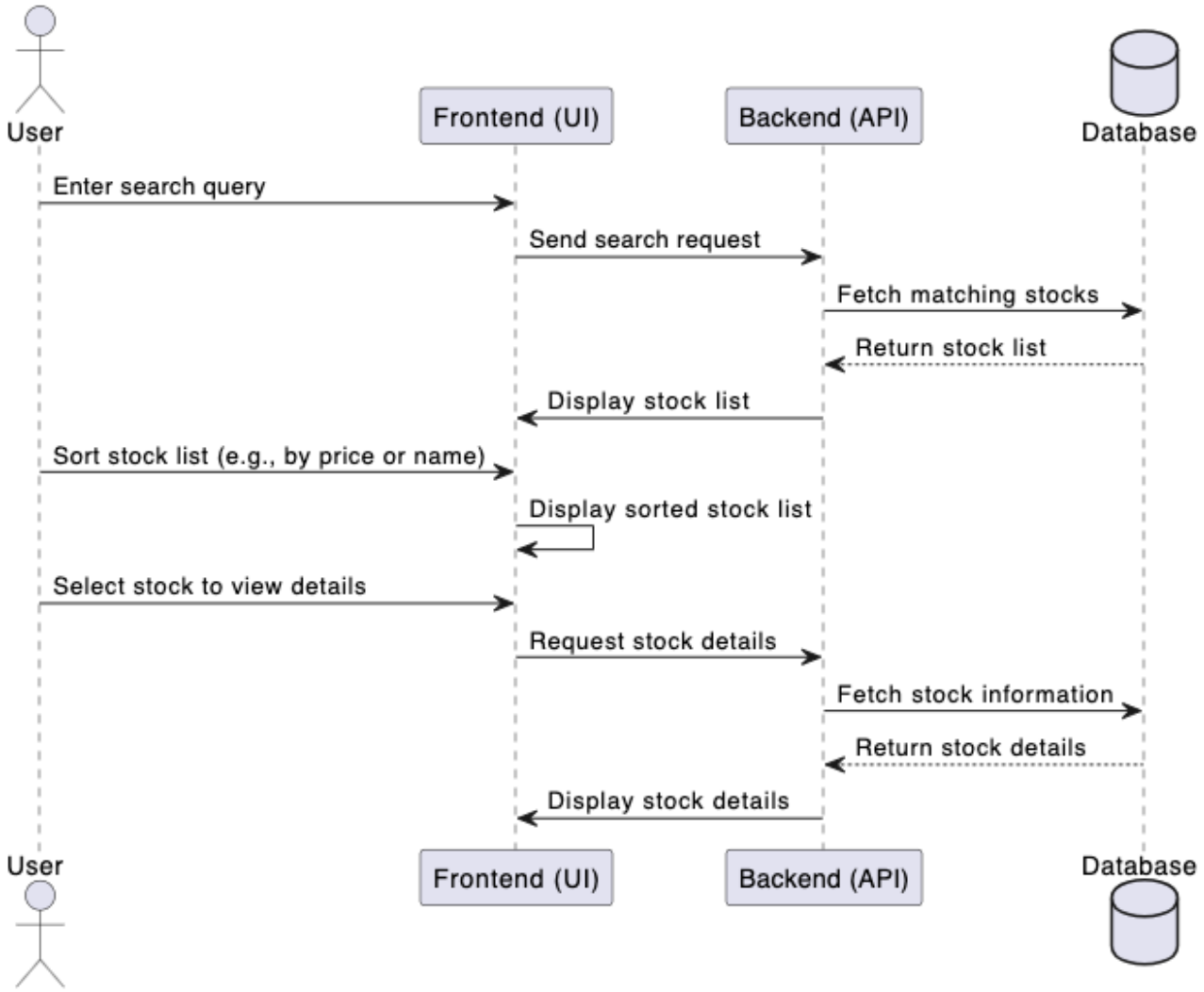

Variables
PASSWORD_SALT : string
OPEN_MARKET_DAY : int
CLOSED_MARKET_DAY : int
OPEN_MARKET_HOURS : int
CLOSED_MARKET_HOURS : int
OPEN_MARKET_MINUTES : int
MINUTE_INTERVAL_BETWEEN_UPDATE : int
TOKEN_VALID_HOURS : int
MAX_ALLOWED_ANALYSIS_PER_DAY : int
DatabaseSettings
DatabaseName : string
StocksCollectionName : string
UsersCollectionName : string
TokensCollectionName : string
MarketTrendsCollectionName : string
SearchResultsCollectionName : string
Secrets
ConnectionString : string = "CONNECTION_STRING"
ConfigurationKeys
AppVariablesSection : string = "AppVariables"
DatabaseSettingsSection : string = "DatabaseSettings"
Finance15Section : string = "YahooFinance15"
Finance127Section : string = "YahooFinance127"
Finance1Section : string = "YahooFinance1"
RealTimeFinanceDataSection : string = "RealTimeFinanceData"
RealTimeFinanceMarketTrends : string = "MarketTrends"
MarketTrend
Id : string?
TrendName : string
LastUpdatedTime : DateTime
StockNews : List<StockNews>
TrendingStocks : DateTime
StockTrend
Symbol : string?
Type : string?
Name : string?
Price : double?
Change : double?
ChangePercent : double?
PreviousClose : double?
PreOrPostMarket : double?
PreOrPostMarketChange : double?
PreOrPostMarketChangePercent : double?
LastUpdateUtc : string?
Currency : string?
Exchange : string?
ExchangeOpen : string?
ExchangeClose : string?
Timezone : string?
CountryCode : string?
StockNews
ArticleTitle : string?
ArticleUrl : string?
ArticlePhotoUrl : string?
Source : string?
PostTimeUtc : string?
StocksInNews : List<StockTrend>
Stock
Id : string
Name : string
Symbol : string
Price : double
FiftyDayAverage : double
TwoHundredDayAverage : double
FiftyTwoWeekRange : string
FiftyTwoWeekLow : double
FiftyTwoWeekHigh : double
AnalystRating : string?
UpdatedTime : DateTime
StockAnalysis : Analysis
StockNotifications : List<StockNotification>
StockAnalysis
Symbol : string
TargetHighPrice : double
TargetLowPrice : double
TargetMeanPrice : double
TargetMedianPrice : double
UpdatedTime : DateTime
User
Id : string?
Password : string?
FirstName : string
LastName : string
Email : string
ConnectionTokens : List<ConnectionToken>
StockNotifications : List<StockNotification>
WatchingStocksByListName : Dictionary<string, Dictionary<string, WatchingStock>>
StockNotification
Id : string?
StockSymbol : string
UserEmail : string
TargetPrice : double
IsTargetBiggerThanOrEqual : bool
ShouldBeNotified : bool
UserCredentials
Email : string
Password : string
UserDetails
FirstName : string
LastName : string
Email : string
Password : string
PasswordUpdateRequest
Email : string
Password : string
StockSearchResult
Symbol : string
Name : string
ExchDisp : string
TypeDisp : string
SearchResult
Id : string
SearchTerm : string
StockSearchResults : List<StockSearchResult>
StockListDetails
UserEmail : string
ListName : string
ShareSale
UserEmail : string
ListName : string
StockSymbol : string
SharePurchaseGuid : string
WatchingStock
Note : string?
PurchaseGuidToShares : Dictionary<string,Share>
WatchingStockAction
Email : string
ListName : string
StockSymbol : string
Note : string?
SharePurchase
UserEmail : string
StockSymbol : string
PurchasingPrice : double
Amount : double
ListName : string
PurchaseDate : DateTime
Share
Id : string
PurchasingPrice : double
Amount : double
PurchaseDate : DateTime

Sequence Diagrams .6.2

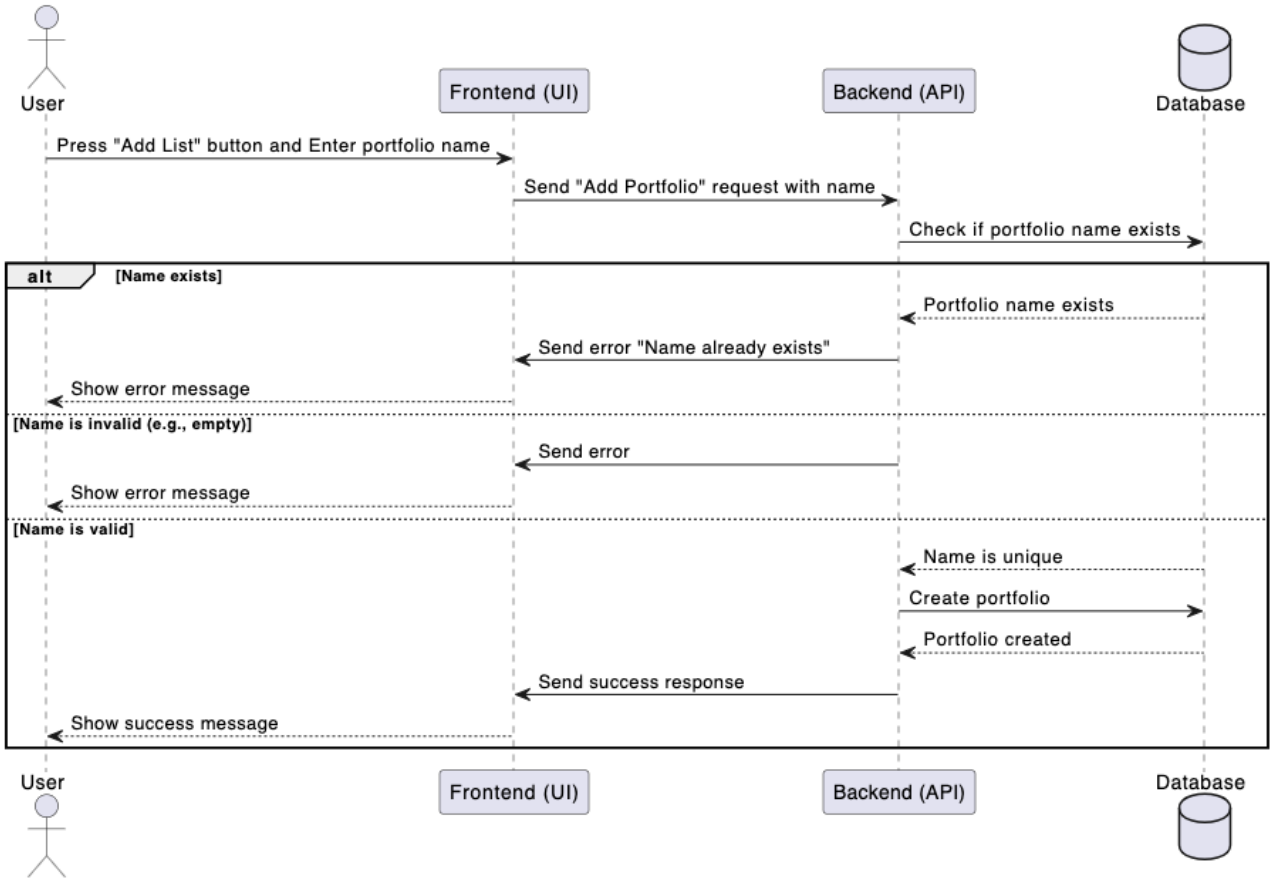
6.2.1. התחברות משתמש



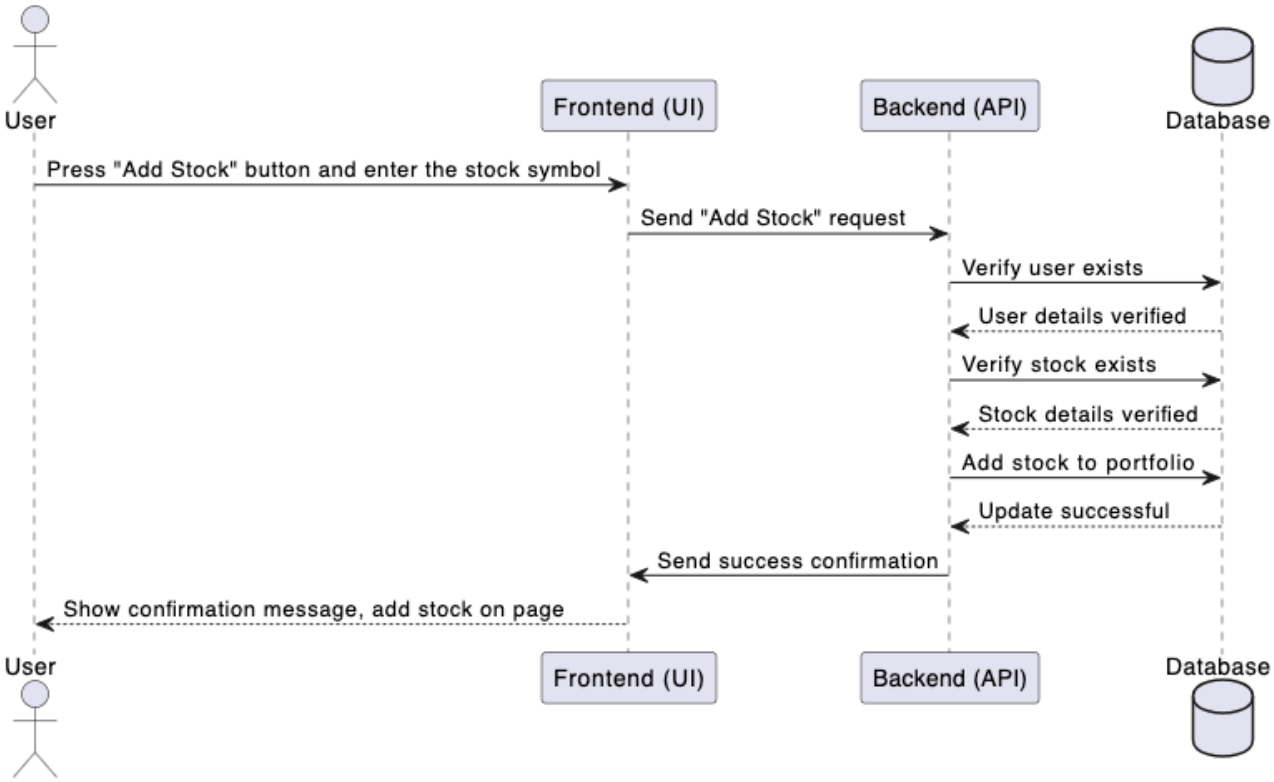
6.2.2. חיפוש מניה



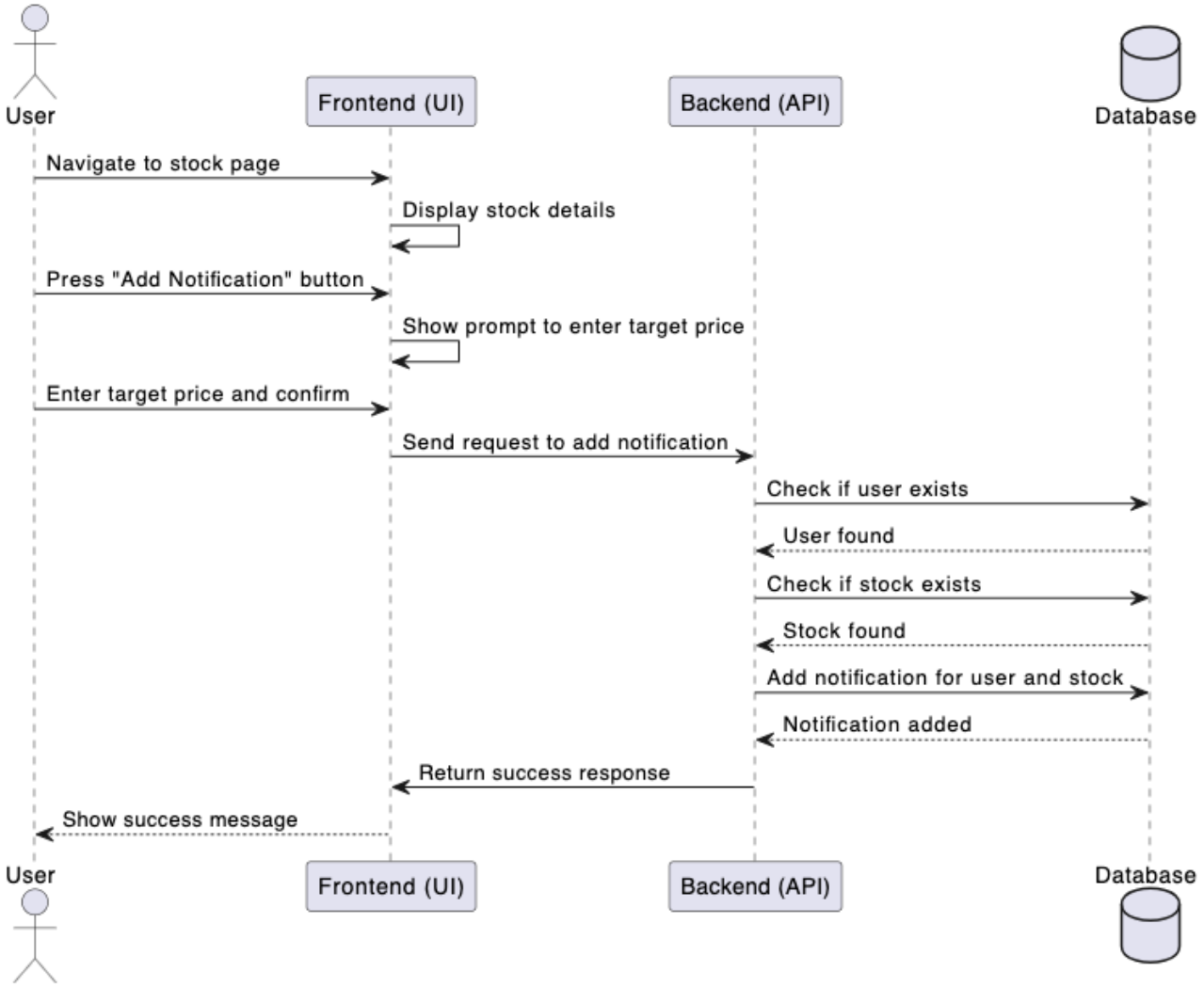
6.2.3. הוספת תיק מניות



6.2.4. הוספת מניה לתיק



6.2.5. הוספת התראה



System Diagram .6.3