‏**תיאור כללי של הסיבה לבניית המערכת**

**סיבה לבניית המערכת:**

המערכת נבנתה במטרה לספק פתרון נוח ויעיל לניהול משימות ופרויקטים אישיים ועסקיים. מערכת ניהול המשימות שלנו מאפשרת למשתמשים לעקוב אחרי המשימות שלהם, לארגן אותן בצורה נוחה ולהבטיח שהן יתבצעו בזמן.

**מה לקוח עשוי להגיד בבקשה לבניית מערכת כזו:**

"אנחנו זקוקים למערכת ניהול משימות שתאפשר לצוות שלנו לעקוב אחרי כל המשימות והפרויקטים בצורה מסודרת ויעילה. אנחנו רוצים שמשתמשים יוכלו ליצור ולערוך משימות, להגדיר תאריכי יעד ולצפות במשימות לפי קטגוריות או פרויקטים. בנוסף, המערכת צריכה לכלול ניהול משתמשים כדי שכל חבר צוות יוכל להירשם, להיכנס למערכת ולנהל את המשימות האישיות שלו או של הצוות. כמו כן, חשוב לנו שתהיה אפשרות לתקשר עם השרת כדי לשמור ולשלוף נתונים בצורה מאובטחת ויעילה."

**תיאור כללי של המערכת:**

**מטרת המערכת:** לספק פתרון מקיף לניהול משימות ופרויקטים, המאפשר למשתמשים ליצור, לערוך, למחוק ולעקוב אחרי משימות אישיות וצוותיות בצורה קלה ונוחה.

**פונקציונליות עיקרית:**

1. **ניהול משימות (ToDo):**
   * **יצירה ועריכה של משימות:** ממשק משתמש גרפי לעריכת פריטים חדשים או קיימים כולל כותרת, תיאור, תאריך יעד, ומצב המשימה.
   * **הצגת משימות:** הצגת כל המשימות בצורה גרפית נעימה, אפשרות לסינון ומיון לפי קטגוריות, פרויקטים, או תאריכי יעד.
   * **ניהול משימות מרובות:** ממשק לעריכה וניהול מספר משימות במקביל.
2. **ניהול משתמשים:**
   * **הרשמה וכניסה למערכת:** אפשרות למשתמשים להירשם למערכת, להיכנס עם שם משתמש וסיסמה, ולשמור את המידע שלהם בצורה מאובטחת.
   * **ניהול פרטי משתמש:** אפשרות למשתמשים לעדכן את הפרטים האישיים שלהם במערכת.
3. **תקשורת עם השרת:**
   * **בקשות HTTP:** המערכת מבצעת בקשות HTTP לשרת לצורך ביצוע פעולות CRUD על נתוני המשימות והמשתמשים.
   * **ניהול אימות:** שימוש בטוקנים של JWT לצורך אימות משתמשים וביצוע פעולות מאובטחות.

המערכת מותאמת למשתמשים פרטיים ועסקיים, המעוניינים לשפר את הארגון והיעילות שלהם בניהול משימות ופרויקטים. היא נבנתה כדי להיות קלה לשימוש, גמישה ומאובטחת, ומתאימה לצרכים של קבוצות עבודה מגוונות.

# 1. `Program.cs`

### פונקציונליות:

- קובץ זה אחראי לנקודת הכניסה הראשית של היישום.

### מטרה:

- לאתחל את הסביבה הגרפית ולהפעיל את הטופס הראשי של היישום.

### שימוש:

- המשתמש פותח את היישום, וטופס המשתמש נטען.

### פעולות:

- אתחול הסביבה הגרפית

- פתיחת הטופס `UserForm`

# 2. `WebClient.cs`

### פונקציונליות:

- לקוח HTTP המבצע בקשות לשרת ה-NodeJS.

### מטרה:

- לתקשר עם השרת לביצוע פעולות CRUD (יצירה, קריאה, עדכון, מחיקה) על נתונים.

### שימוש:

- המשתמש מבצע פעולות באפליקציה כמו כניסה למערכת, יצירה או עריכה של פריטים, והלקוח שולח בקשות לשרת בהתאם.

### פעולות:

- ביצוע בקשות GET, POST, PUT, DELETE לשרת

- טיפול באימות והוספת טוקן JWT לבקשות

\*מתודות ודברים שעושה השרת יופיעו בהמשך

# `ToDo`

## `BubbleControl.cs`

### פונקציונליות:

- רכיב UI להצגת פריטים של Todo בצורה ויזואלית מושכת.

### מטרה:

- לשפר את חוויית המשתמש על ידי הצגת הפריטים בצורה גרפית.

### שימוש:

- המשתמש רואה את הפריטים המוצגים ברשימה או בלוח.

### פעולות:

- הצגת פריטים בודדים

- שינוי מצב הפריט (סימון כהושלם וכדומה)

## `SingleToDoEdit.cs`

### פונקציונליות:

- טופס או רכיב לעריכת פריט Todo בודד.

### מטרה:

- לאפשר למשתמש לערוך את פרטי הפריט.

### שימוש:

- המשתמש פותח את הטופס כדי לערוך פריט קיים או ליצור חדש.

### פעולות:

- עריכת כותרת, תיאור, תאריך יעד וכדומה

- שמירה ועדכון של הפריט

## `ToDoModel.cs`

### פונקציונליות:

- מודל נתונים לפריטי Todo.

### מטרה:

- להגדיר את מבנה הנתונים של פריט Todo.

### שימוש:

- המערכת משתמשת במודל זה ליצירה, קריאה, עדכון ומחיקה של פריטי Todo.

### פעולות:

- הגדרת שדות כמו מזהה, כותרת, תיאור, תאריך יעד, מצב וכדומה

## `ToDosEditor.cs`

### פונקציונליות:

- טופס או רכיב לעריכת פריטי Todo מרובים.

### מטרה:

- לאפשר למשתמש לנהל מספר פריטים במקביל.

### שימוש:

- המשתמש רואה רשימה או לוח של פריטי Todo ויכול לערוך אותם.

### פעולות:

- הצגה ועריכה של רשימה של פריטים

- ביצוע פעולות גורפות כמו מחיקה או סימון כהושלמים

## `ToDoService.cs`

### פונקציונליות:

- לוגיקת העסק הקשורה לפריטי Todo.

### מטרה:

- לטפל בבקשות לשרת ולנהל את פריטי ה-Todo.

### שימוש:

- המערכת משתמשת בשירות זה ליצירה, קריאה, עדכון ומחיקה של פריטים.

### פעולות:

- שליחה וקבלה של נתונים משרת ה-NodeJS

- ניהול פריטי Todo במערכת

# `User`

## `UserForm.cs`

### פונקציונליות:

- טופס ממשק משתמש לפעולות הקשורות למשתמשים כמו כניסה, הרשמה וניהול פרופיל.

### מטרה:

- לספק ממשק גרפי לניהול משתמשים.

### שימוש:

- המשתמש נכנס למערכת, נרשם או מעדכן את הפרטים שלו.

### פעולות:

- הצגת טופס כניסה או הרשמה

- שליחה וקבלה של נתוני משתמש לשרת

## `UserModel.cs`

### פונקציונליות:

- מודל נתונים למידע על משתמשים.

### מטרה:

- להגדיר את מבנה הנתונים של משתמש.

### שימוש:

- המערכת משתמשת במודל זה לניהול פרטי המשתמשים.

### פעולות:

- הגדרת שדות כמו מזהה, שם, אימייל, סיסמה, טוקן וכדומה

## `UserService.cs`

### פונקציונליות:

- לוגיקת העסק הקשורה למשתמשים.

### מטרה:

- לטפל בבקשות לשרת ולנהל את המשתמשים.

### שימוש:

- המערכת משתמשת בשירות זה לכניסה, הרשמה ועדכון פרטי משתמש.

### פעולות:

- שליחה וקבלה של נתונים משרת ה-NodeJS

- ניהול פרטי המשתמשים במערכת

## `UserSession.cs`

### פונקציונליות:

- ניהול מידע על המושב הנוכחי של המשתמש.

### מטרה:

- לשמור מידע על ההתחברות הנוכחית של המשתמש.

### שימוש:

- המערכת משתמשת במחלקה זו לשמירת טוקן ההתחברות ומידע נוסף על המושב הנוכחי.

### פעולות:

- שמירת טוקן התחברות

- אחזור מידע על המשתמש המחובר

# איך בנוי השרת ומה הוא עושה?

To-Do List Application API Documentation

Overview

This document outlines the API endpoints available for the To-Do List Application.

It provides information on the methods used to create, retrieve, update, and manage to-do items.

API Methods

Enums:

TodoStatus {

“NOTE”,

“TODO”,

“IN\_PROGRESS”,

“COMPLETED”,

“DELETED”,

“NOTE”  
}

**Signup**

Endpoint Description:

Sign up new user

HTTP Method: POST

URI: /api/user/signup

Request Payload: json

{

 "email": "string",

 "password": "string",

 "nickname": "string",

}

Returns  
{  
“token”: string  
}

**Login**

Endpoint Description:

Logout user

HTTP Method: POST

URI: /api/user/login

Request Payload: None

Error Codes:

401 Unauthorized: User not authenticated.

500 Internal Server Error: An error occurred on the server.

**CreateToDoItem**

Endpoint Description:

Creates a new to-do item with the specified details.

HTTP Method: POST

URI: /api/todo/:token

Request Payload: json

{

 "title": "string",

 "description": "string",

 "parentTaskId": "number",

 "startDate": "YYYY-MM-DDTHH:MM:SS",

 "dueDate": "YYYY-MM-DDTHH:MM:SS",

 "endDate": "YYYY-MM-DDTHH:MM:SS",

 "assignee": "number",

 "status": TodoStatus

}

Response Structure: json

{

 "id": "number",

 "title": "string",

 "description": "string",

 "parentTaskId": "number",

 "startDate": "YYYY-MM-DDTHH:MM:SS",

 "dueDate": "YYYY-MM-DDTHH:MM:SS",

 "endDate": "YYYY-MM-DDTHH:MM:SS",

 "assignee": "number",

 "status": TodoStatus

}

Error Codes:

400 Bad Request: Missing or invalid request payload.

401 Unauthorized: User not authenticated.

500 Internal Server Error: An error occurred on the server.

Operation Example:

POST /api/todoitems/:token HTTP/1.1

Host: example.com

Content-Type: application/json

{

 "title": "Buy milk",

 "description": "2 liters of skim milk",

 "parentTaskId": "",

 "startDate": "2023-04-01T09:00:00",

 "dueDate": "2023-04-02T18:00:00",

 "endDate": "2023-04-02T18:00:00",

 "assignee": "john.doe@example.com",

 "status":TodoStatus

}

**UpdateToDoItem**

Endpoint Description:

Updates an existing to-do item.

HTTP Method: PUT

URI: /api/todoitems/{Id}/:token

Request Payload:

{

 "title": "string (optional)",

 "description": "string (optional)",

 "parentTaskId": "number (optional)",

 "startDate": "YYYY-MM-DDTHH:MM:SS (optional)",

 "dueDate": "YYYY-MM-DDTHH:MM:SS (optional)",

 "endDate": "YYYY-MM-DDTHH:MM:SS (optional)",

 "assignee": "number (optional)",

 "status": TodoStatus

}

Response Structure:

{

 "Id": "number",

 "title": "string",

 "description": "string",

 "parentTaskId": "number",

 "startDate": "YYYY-MM-DDTHH:MM:SS",

 "dueDate": "YYYY-MM-DDTHH:MM:SS",

 "endDate": "YYYY-MM-DDTHH:MM:SS",

 "assignee": "number",

 "status": TodoStatus

}

Error Codes:

400 Bad Request: Missing or invalid fields in the request payload.

404 Not Found: The specified to-do item Id was not found.

401 Unauthorized: User not authenticated.

**DeleteToDoItemById**

Endpoint Description:

Deletes a to-do item by its Id.

HTTP Method: DELETE

URI: /api/todoitems/{Id}/:token

Request Payload: None

Response Structure: None

Error Codes:

404 Not Found: The specified to-do item Id was not found.

401 Unauthorized: User not authenticated.

**AssignTaskToUser**

Endpoint Description:

Assigns an existing to-do item to a specified user.

HTTP Method: PUT

URI: /api/todoitems/assign/{assigneeId}/:token

Request Payload:

{

 "assignee": "number"

}

Response Structure:

{

 "key": "number",

 "title": "string",

 "description": "string",

 "parentTaskKey": "number",

 "startDate": "YYYY-MM-DDTHH:MM:SS",

 "dueDate": "YYYY-MM-DDTHH:MM:SS",

 "endDate": "YYYY-MM-DDTHH:MM:SS",

 "assignee": "number",

 "status": TodoStatus

}

Error Codes:

400 Bad Request: Missing or invalid request payload or the specified key does not exist.

401 Unauthorized: User not authenticated.

403 Forbidden: User does not have permission to assign tasks.

404 Not Found: The specified to-do item key was not found.

500 Internal Server Error: An error occurred on the server.

Operation Example:

PUT /api/todoitems/assign/230401123456 HTTP/1.1

Host: example.com

Content-Type: application/json

{

 "assignee": "jane.doe@example.com"

}

**GetToDoItemsByAssignee**

Endpoint Description:

Retrieves all to-do items assigned to a specific user.

HTTP Method: GET

URI: /api/todoitems/assignee/{assignee}/:token

Request Payload: None

Response Structure:

[

 {

 "key": "number",

 "title": "string",

 "description": "string",

 "parentTaskKey": "string",

 "startDate": "YYYY-MM-DDTHH:MM:SS",

 "dueDate": "YYYY-MM-DDTHH:MM:SS",

 "endDate": "YYYY-MM-DDTHH:MM:SS",

 "assignee": "string",

 "completed": boolean

 }

]

Error Codes:

401 Unauthorized: User not authenticated.

**GetToDoItemById**

Endpoint Description:

Retrieves a to-do item by its key.

HTTP Method: GET

URI: /api/todoitems/{key}/:token

Request Payload: None

Response Structure:

{

 "key": "number",

 "title": "string",

 "description": "string",

 "parentTaskKey": "string",

 "startDate": "YYYY-MM-DDTHH:MM:SS",

 "dueDate": "YYYY-MM-DDTHH:MM:SS",

 "endDate": "YYYY-MM-DDTHH:MM:SS",

 "assignee": "number",

 "completed": boolean

}

Error Codes:

404 Not Found: The specified to-do item key was not found.

401 Unauthorized: User not authenticated.

**GetAllToDoItems**

Endpoint Description:

Retrieves all to-do items.

HTTP Method: GET

URI: /api/todoitems/:token

Request Payload: None

Response Structure:

[

 {

 "key": "number",

 "title": "string",

 "description": "string",

 "parentTaskKey": "number",

 "startDate": "YYYY-MM-DDTHH:MM:SS",

 "dueDate": "YYYY-MM-DDTHH:MM:SS",

 "endDate": "YYYY-MM-DDTHH:MM:SS",

 "assignee": "number",

 "status": TodoStatus

 }

]

Error Codes:

401 Unauthorized: User not authenticated.

**ChangeTodoStatus**

Endpoint Description:

Marks a to-do item as completed.

HTTP Method: PUT

URI: /api/todoitem/{Id}?status={newStatus}

Request Payload: None

Response Structure:

{

 "Id": "number",

 "status": TodoStatus

}

Error Codes:

404 Not Found: The specified to-do item Id was not found.

401 Unauthorized: User not authenticated.

**GetToDoItemsForDate**

Endpoint Description:

Retrieves all to-do items due today.

HTTP Method: GET

URI: /api/todoitems/date/{date}/:token

Request Payload: None

Response Structure:

[

 {

 "key": "string",

 "title": "string",

 "description": "string",

 "parentTaskKey": "string",

 "startDate": "YYYY-MM-DDTHH:MM:SS",

 "dueDate": "YYYY-MM-DDTHH:MM:SS",

 "endDate": "YYYY-MM-DDTHH:MM:SS",

 "assignee": "string",

 "completed": boolean

 }

]

Error Codes:

401 Unauthorized: User not authenticated.

**GetOverdueToDoItems**

Endpoint Description:

Retrieves all overdue to-do items.

HTTP Method: GET

URI: /api/todoitems/overdue/:token

Request Payload: None

Response Structure:

[

 {

 "key": "string",

 "title": "string",

 "description": "string",

 "parentTaskKey": "string",

 "startDate": "YYYY-MM-DDTHH:MM:SS",

 "dueDate": "YYYY-MM-DDTHH:MM:SS",

 "endDate": "YYYY-MM-DDTHH:MM:SS",

 "assignee": "string",

 "completed": boolean

 }

]

Error Codes:

401 Unauthorized: User not authenticated.

**GetAllAssigns**

Endpoint Description:

Gets all possible assignees as id+name

HTTP Method: GET

URI: /api/user/assigns/:token

Request Payload

Response Structure:

[

 {

 “Id” :number

“Name”: string

 }

]

Error Codes:

401 Unauthorized: User not authenticated.

# סיכום כללי

הפרויקט שלנו הוא מערכת לניהול פריטי רשימת משימות עם ממשק משתמש נוח שמאפשר למשתמשים לנהל את המשימות שלהם ולבצע פעולות כמו יצירה, עריכה, ומחיקה של פריטים. המערכת כוללת פונקציונליות של ניהול משתמשים הכוללת כניסה, הרשמה, וניהול פרטי משתמשים. כל הנתונים מנוהלים בעזרת שרת NodeJS שמבצע את פעולות ה-CRUD.