Smart Scheduler App - API

# Introduction

* סוגי משימות: Retrieve (single, all), Create, Update, Delete
* אובייקטים: Task, TeamMember, Assignment
* Skills:  
  בצד הלקוח ממומשת מחלקת עזר SkillRepository המכילה רשימה סגורה של כישורים לבחירה.  
  הכישורים יהיו ניתנים לבחירה מתוך רשימה נפתחת דינמית ויישלחו כ List<String> לשרת.
* בצד לקוח: בכל שינוי (הוספה/עדכון/מחיקה) לבצע Refresh לנתונים.
* כל הפונקציות צריכות להחזיר תשובה כדי לדעת האם המשימה צלחה או לא.
* צריך לטפל בכל מקרי הקצה.

# Task

**getAllTasks()**

* שולף מהשרת את כל המשימות ושולח ללקוח.

**getTaskById()**

* מחפשים במאגר המידע את המשימה לפי ID ומחזירים אותה.

**CreateNewTask()**

* יצירת ID ייחודי לפי מנגנון אוטומטי
* ולידציה לנתונים ברמת לקוח
* ולידציה לנתונים ברמת controller
* Skills כ- List<String> המגיעים מתוך רשימה סגורה

**updateTask()**

* פרטים משתנים, הוספה/החסרה של נתונים (skills, Priority, durationHours).
* משפיע על הקצאת המשימות לכלל העובדים (נניח שדורשת עוד skill, יש יותר שעות וכו').
* עדכון משימה >> הקצאת משימות חדשה לכלל העובדים (assignTasks).

**deleteTask()**

* אם המשימה הוקצתה, צריך למחוק הקצאה (deleteAssignment)

# TeamMember

**getAllTeamMembers()**

* שולף מהשרת את כל העובדים ושולח ללקוח.

**getTeamMemberById()**

* מחפשים במאגר המידע את העובד לפי ID ומחזירים אותו.

**createNewTeamMember()**

* יצירת ID ייחודי לפי מנגנון אוטומטי
* ולידציה לנתונים ברמת לקוח
* ולידציה לנתונים ברמת controller
* Skills כ- List<String> המגיעים מתוך רשימה סגורה

**updateTeamMember()**

* פרטים משתנים, הוספה/החסרה של נתונים (skills, MaxHoursPerDay, efficiency).
* משפיע על הקצאת המשימות לכלל העובדים (נוספה/נגרעה יכולת, נחסרו שעות עבודה וכו').
* עדכון עובד >> הקצאת משימות חדשה לכלל העובדים (assignTasks).

**deleteTeamMember()**

* אם העובד הוקצה למשימה, צריך למחוק הקצאה (deleteAssignment).

# Assignment

**assignTasks(ITaskAssignment strategy)**

* ליבת הפרויקט, החלק של החישוב החכם.
* טען את רשימת המשימות ורשימת העובדים מהזיכרון.
* מחק את כל ההקצאות הקודמות (AssignmentDao.deleteAll()).
* חשב הקצאות לכולם לפי strategy (greedy/balanced)
* שמור את ההקצאות החדשות בזיכרון (assignmentDao.save(assignments)))

**assignTasksToTeamMember(TeamMember ID, strategy)**

* ליבת הפרויקט, החלק של החישוב החכם.
* טען את רשימת המשימות בלבד מהזיכרון.
* השאר את כל ההקצאות הקיימות בזיכרון.
* חשב הקצאות לעובד הספציפי לפי strategy (greedy/balanced).
* אם הקצאה כבר קיימת בזיכרון, עדכן את שעות ההקצאה (updateAssignment).
* הוסף את ההקצאות החדשות לזיכרון (assignmentDao.save(assignments))).

**הסבר:**

* הקצאת המשימות הכוללת תהיה מדשבורד (או משהו אחר בסגנון) שבו רואים את כל העובדים/משימות. בוחרים שיטה להקצאה (greedy/balanced) ואז מריצים לכולם. ההרצה הזאת תדרוס את ההקצאות הקודמות.
* הקצאת המשימה הפרטנית לעובד תהיה בדשבורד/תפריט לעובד ספציפי. יכול להיות שפריסת המשימות לא תהיה אופטימלית במצב הזה.
* הנחת היסוד:
  + בהקצאת משימה כוללת מקבלים חלוקת משימות אופטימלית לפי האלגוריתם.
  + בהקצאת משימה פרטנית מניחים שכבר כולם קיבלו הקצאות עבודה ולא שייך לדרוס לכולם רק בגלל עובד חדש שהגיע עכשיו.

**getAllAssignments()**

* שולף מהשרת את כל ההקצאות ושולח ללקוח.

**deleteAssignment(String taskId, String memberId)**

* מחק משימה (assignmentDao.deleteById(id)).
* הוסף assignedHours לעובד ולמשימה.
* החזר תשובה לקריאה האם צלח או לא.

# Statistics

ליבת הפרויקט זה להראות שינוי בין בחירה greedy vs balanced

אלו הסטטיסטיקות שמומשו:

**countTasks()**

* מחזירה את כמות המשימות הכוללת.

**countTeamMembers()**

* מחזירה את כמות העובדים הכוללת.

**countUnassignedTasks()**

* מחזירה את כמות המשימות שאין להן הקצאות בכלל.

**averageLoad()**

* מחזירה את אחוז העומס הממוצע על העובדים (כדי לראות שאכן האלגוריתמים עובדים).

**searchTasksByName()**

* מחפשת משימה לפי השם שלה ומחזירה את האובייקט שלה.

**searchTeamMembersByName()**

* מחפשת עובד לפי השם שלו ומחזירה את האובייקט שלו.