

מטלה - שיעור 8

ד"ר יורם סגל

בנייה מערכת עם ארכיטקטורת סוכני AI mbosst MCP Server

1 מבוא

מטרת המטלה היא לישם את עקרונות הארכיטקטורה שנלמדו בשיעור, תוך בניית מערכת תוכנה מודולרית ומקצועית.

חופש יצירתי

המטלה מוגדרת בצורה פתוחה. הסטודנט רשאי לבחור את הפרויקט שלו ולפרש את הדרישות לפי הבנותו, כל עוד הוא מדגים את עקרונות הארכיטקטורה שנלמדו.

2 נושא הפרויקט

1.2 אפשרויות מומלצת: מערכת חשבוניות

מערכת להכנת חשבוניות הכוללת:

- ניהול לקוחות
- הפיקת חשבוניות (חשבונית מס, חשבונית עסקה, קבלה)
- ניהול מספרים סידוריים
- חישובי מע"מ
- שליחת חשבוניות

2.2 אפשרות חלופית

הסטודנט רשאי לבחור פרויקט אחר לפי העדפתו, בתנאי שהפרויקט מאפשר הדגמה של כל עקרונות הארכיטקטורה הנדרשים.

3 חממת השלבים

יש לבנות את הפרויקט בחמשה שלבים מתפתחים. כל שלב מבוסס על הקודם ומוסיף רכיבים נוספים.

1.3 שלב 1: תשתיות בסיסיות

מטרה: הקמת שכבה התשתיות והshitof
רכיבים לדוגמה:

- שכבת קונפיגורציה (Configuration)
- מנגנון לוגים (Logging)
- טיפול בשגיאות (Exception Handling)
- מבנה תיקיות הפרויקט
- בדיקות ייחודicas בסיסיות

פתחות לפתרונות: הסטודנט יגדיר אילו תשתיות נדרשות למערכת שבחר.

2.3 שלב 2: שרת MCP עם כלים

מטרה: בניית שרת MCP בסיסי עם Tools
רכיבים לדוגמה:

- הגדרת כלים (Tools) עם מזהה, תיאור וסכמה JSON
- לוגיקה עסקית בסיסית
- חיבור לאחסון נתונים

פתחות לפתרונות: הסטודנט יבחר אילו כלים למשולש ומה תהיה הפונקציונליות שלהם.

3.3 שלב 3: שלושת הפרימיטיבים

מטרה: הרחבת המערכת לתמיכה בכל הפרימיטיבים של MCP
רכיבים:

- Tools - פעולות שימוש מצב (Write)
- Resources - קריאת נתונים (Read) - סטטיים ודינמיים
- Prompts - תבניות להדרכת המודל

פתחות לפתרונות: הסטודנט יגדיר אילו משאבים ותבניות נדרשים למערכת שלו.

4.3 שלב 4: שכבת תקשורת

מטרה: הוספת שכבת תקשורת מודולרית
רכיבים לזוגמהה:

- הפרדת שכבת התקשרות מהלוגיקה
- תמייה לפחות סוג תקשורת אחד (למשל: HTTP/SSE, STDIO)
- יכולת להחליף את שכבת התקשרות ללא שינוי הקוד

פתחות לפתרונות: הסטודנט יבחר את שיטת התקשרות המתאימה לפרויקט שלו.

5.3 שלב 5: SDK וממשק משתמש

מטרה: הוספת שכבת SDK ולפחות שני סוגי ממוקש משתמש משתמש
רכיבים:

- שכבת SDK - כל הפעולות עוברות דרך
- לפחות שני סוגי GUI (בחירה):
 - ממוקש טרמינל (CLI)
 - אפליקציית דסקטופ
 - ממוקש ווב (Web)
 - אחר

פתחות לפתרונות: הסטודנט יבחר אילו ממוקשים למעשה ומה תהיה חווית המשתמש.

4 עקרונות ארכיטקטורה נדרשים

יש להקפיד על העקרונות הבאים לאורך כל הפרויקט:

1. **אפס ערכים קבועים (No Hard Coding)** - כל הערכים מגיעים מIRONConfiguraTION
2. **מודולריות** - כל רכיב ניתן להחלפה ללא שינוי שאר המערכת
3. **קבצים קצריים** - מומלץ עד 150 שורות לקובץ
4. **אפס שפוף קוד** - קוד חוזר על עצמו צריך להיות בפונקציה או קלאס
5. **עבודה מונחת עצמים (OOP)** - שימוש בklassים לארגון הקוד
6. **בדיקות יחידה (Unit Tests)** - בדיקות לכל רכיב
7. **טיפול במשאים** - מנגנון נעליה, ניסיון חוזר וטיפול בשגיאות

5 הגשה

1.5 מבנה ההגשה

- אחד הכלול את כל חמישת השלבים Repository
- כל שלב כפרויקט/ענף נפרד בתוך ה-Repository
- כל שלב מבוסס על הקודם ומוסיף לו

טייפ לעובודה מקבילית

לעבודה יעלה עם מספר סוכני AI במקביל, מומלץ להשתמש ב-Git Worktrees עם דפוס ``ענף לכל סוכן''. ראו נספח א' לפרטים נוספים.

2.5 מה להגיש

- קוד מקור מלא
- בדיקות יחידה
- תיעוד בסיסי (לפי שיקול הסטודנט)

6 הערכה

הערכתה תתבסס על:

- הדגמת עקרונות הארכיטקטורה שנלמדו
- מודולריות והפרדה בין שכבות
- איקות הקוד והבדיקות
- התקזימות ברורה בין השלבים

הערה

המטרה העיקרית היא ללמידה לנוכח דרישות בפרומפט וליצור ארכיטקטורה נכונה. ברגע שהארכיטקטורה נכונה, אפשר ליצור מערכות במהירות גבוהה.

A' פיתוח מקבילי עם Git Worktrees

A'.1 הבעיה: עבודה במקביל

כשסוכני AI נוכנים לתמונה, האתגר הופך למורכב: כיצד ניתן למסוף סוכנים חכמים לעבוד על אותו קוד בו-זמן?

A'.2 מה זה ?Worktree

הגדרה
Git Worktree הוא מנגן שמאפשר לבדוק (checkout) מספר ענפים מאותו מאגר לתיקות נפרדות בו-זמן. כל Worktree הוא עצמאי לחלוטין מבחינות קבצים, אך משותף ההיסטוריה ומקור עם כל האחרים.

A'.3 דפוס ``ענף לכל סוכן''

提议在每个 Worktree 下面创建一个单独的分支，以实现并发开发。

- **בידוד מלא** --- כל סוכן עובד על קבצים משלו
- **היסטוריה משותפת** --- קל למזג את העבודה
- **עבודה מקבילתית אמיתית** --- סוכנים עובדים בו-זמן
- **ניהול פשוט** --- פקודות Git רגילות

A'.4 מדריך מעשי

A'.4.1 שלב 1: יצרת Worktrees

בינכוס השולשל Worktrees תריצי

```
# Navigate to main project
cd /path/to/project

# Create worktree for Agent 1: Feature Development
git worktree add ../project-agent1-feature -b agent1/new-api

# Create worktree for Agent 2: Bug Fix
git worktree add ../project-agent2-bugfix -b agent2/fix-security

# Create worktree for Agent 3: Documentation
git worktree add ../project-agent3-docs -b agent3/update-readme

# Verify all worktrees exist
git worktree list
```

א' 2.4. שלב 2: הפעלת הסוכנים

בכל טרמינל נפרד, הפעילו סוכן Claude Code עם משימה ייוזדית:

ליבקאמ בינוכס תלעפה

```
# Terminal 1: Agent 1 - Feature Development
cd ../project-agent1-feature
claude
> "Implement OAuth2 authentication for the API"

# Terminal 2: Agent 2 - Bug Fix
cd ../project-agent2-bugfix
claude
> "Fix the SQL injection vulnerability in user login"

# Terminal 3: Agent 3 - Documentation
cd ../project-agent3-docs
claude
> "Update README with new API documentation"
```

א' 3.4. שלב 3: מיזוג העבודה

כשוכן מסיים, ממזגים את הענף שלו ל-main:

бинוכסה תדובע גוזים

```
# Return to main project
cd /path/to/project

# Merge Agent 1's feature branch
git merge agent1/new-api -m "Feature: OAuth2 authentication"

# Merge Agent 2's bugfix branch
git merge agent2/fix-security -m "Fix: SQL injection
vulnerability"

# Clean up worktrees
git worktree remove ../project-agent1-feature
git worktree remove ../project-agent2-bugfix
```

א' 5. טיפים מתקדמים

- **שמות ענפים תיאוריים:** agent2/bugfix-security ,agent1/feature-oauth
- **סנכרון:** השתמשו ב-git rebase origin/main לעדכן הענף
- **בדיקה קונפליקטיבית:** git merge --no-commit --no-ff לבדיקה לפני מיזוג

העתיד של פיתוח תוכנה הוא עבודה במקביל עם סוכנים חכמים.

© כל הזכויות שמורות לד"ר יoram סגל