

מטלה - שיעור 8

ד"ר יורם סגל

בניית מערכת עם ארכיטקטורת סוכני AI

מבוססת MCP Server

1 מבוא

מטרת המטלה היא ליישם 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 הבנתו, כל עוד הוא מדגים את עקרונות הארכיטקטורה שנלמדו.

2 נושא הפרויקט

1.2 אפשרות מומלצת: מערכת חשבוניות

מערכת להכנת חשבוניות הכוללת:

- ניהול לקוחות
- הפקת 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 חמשת השלבים

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

1.3 שלב 1: תשתיות בסיסיות

מטרה: הקמת שכבת התשתיות והשיתוף
רכיבים לדוגמה:

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

פתוח לפרשנות: 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 וסכמת JSON
- לוגיקה עסקית בסיסית
- חיבור לאחסון נתונים

פתוח לפרשנות: הסטודנט יבחר אילו כלים לממש ומה תהיה הפונקציונליות שלהם.

3.3 שלב 3: שלושת הפרימיטיבים

מטרה: הרחבת המערכת לתמיכה בכל הפרימיטיבים של MCP
רכיבים:

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

פתוח לפרשנות: הסטודנט יגדיר אילו משאבים ותבניות נדרשים למערכת שלו.

4.3 שלב 4: שכבת תקשורת

מטרה: הוספת שכבת תקשורת מודולרית
רכיבים לדוגמה:

- הפרדת שכבת התקשורת מהלוגיקה
 - תמיכה בלפחות סוג תקשורת אחד (למשל: HTTP/SSE, STDIO)
 - יכולת להחליף את שכבת התקשורת ללא שינוי הקוד
- פתוח לפרשנות:** הסטודנט יבחר את שיטת התקשורת המתאימה לפרויקט שלו.

5.3 שלב 5: SDK וממשק משתמש

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

- שכבת SDK - כל הפעולות עוברות דרכה
 - לפחות שני סוגי GUI (לבחירה):
 - ממשק טרמינל (CLI)
 - אפליקציית דסקטופ
 - ממשק ווב (Web)
 - אחר
- פתוח לפרשנות:** הסטודנט יבחר אילו ממשקים לממש ומה תהיה חוויית המשתמש.

4 עקרונות ארכיטקטורה נדרשים

יש להקפיד על העקרונות הבאים לאורך כל הפרויקט:

1. **אפס ערכים קבועים (No Hard Coding)** - כל הערכים מגיעים מקונפיגורציה
2. **מודולריות** - כל רכיב ניתן להחלפה ללא שינוי שאר המערכת
3. **קבצים קצרים** - מומלץ עד 150 שורות לקובץ
4. **אפס שכפול קוד** - קוד שחוזר על עצמו צריך להיות בפונקציה או קלאס
5. **עבודה מונחית עצמים (OOP)** - שימוש בקלאסים לארגון הקוד
6. **בדיקות יחידה (Unit Tests)** - בדיקות לכל רכיב
7. **טיפול במשאבים** - מנגנוני נעילה, ניסיון חוזר וטיפול בשגיאות

5 הגשה

1.5 מבנה ההגשה

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

טיפ לעבודה מקבילית

לעבודה יעילה 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

ההערכה תתבסס על:

- הדגמת עקרונות הארכיטקטורה שנלמדו
- מודולריות והפרדה בין שכבות
- איכות הקוד והבדיקות
- התקדמות ברורה בין השלבים

הערה

המטרה העיקרית היא ללמוד לנסח דרישות בפרומפט ולייצר ארכיטקטורה נכונה. ברגע שהארכיטקטורה נכונה, אפשר לייצר מערכות במהירות גבוהה.

א' פיתוח מקבילי עם Git Worktrees

א.1 הבעיה: עבודה במקביל

כשסוכני AI נכנסים לתמונה, האתגר הופך למורכב: כיצד נאפשר למספר סוכנים חכמים לעבוד על אותו קוד בו-זמנית?

א.2 מה זה Worktree?

הגדרה

Git Worktree הוא מנגנון שמאפשר לבדוק (checkout) מספר ענפים מאותו מאגר לתיקיות נפרדות בו-זמנית. כל Worktree הוא עצמאי לחלוטין מבחינת קבצים, אך משתף היסטוריה ומקור עם כל האחרים.

א.3 דפוס ``ענף לכל סוכן''

Anthropic ממליצה לתת לכל סוכן Worktree משלו. היתרונות:

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

א.4 מדריך מעשי

א.1.4 שלב 1: יצירת Worktrees

מינוס 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 ייעודית:

ליבקמב פינכוס 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

כשסוכן מסיים, ממזגים את הענף שלו ל-main:

פינכוסה תדובע גוזימ

```
# 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: agent1/feature-oauth, agent2/bugfix-security
- סנכרון: השתמשו ב-git rebase origin/main לעדכון הענף
- בדיקת קונפליקטים: git merge --no-commit --no-ff לבדיקה לפני מיזוג

העתיד של פיתוח תוכנה הוא עבודה במקביל עם סוכנים חכמים.

© כל הזכויות שמורות לד"ר יורם סגל