**סקר ספרות ודרישות:**

חלק 1: סקר מצב קיים.

מערכות דומות:

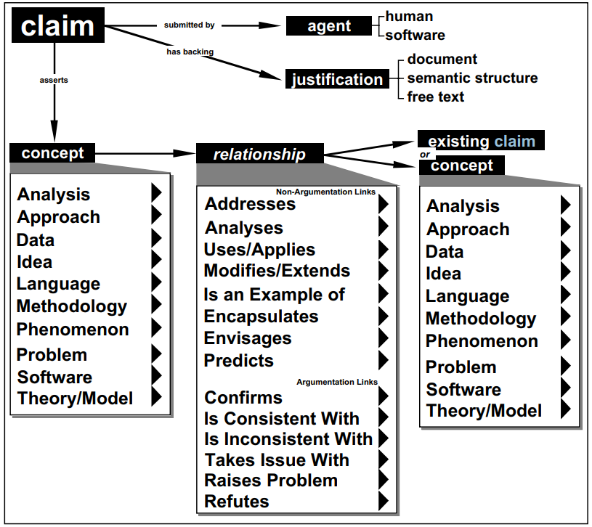
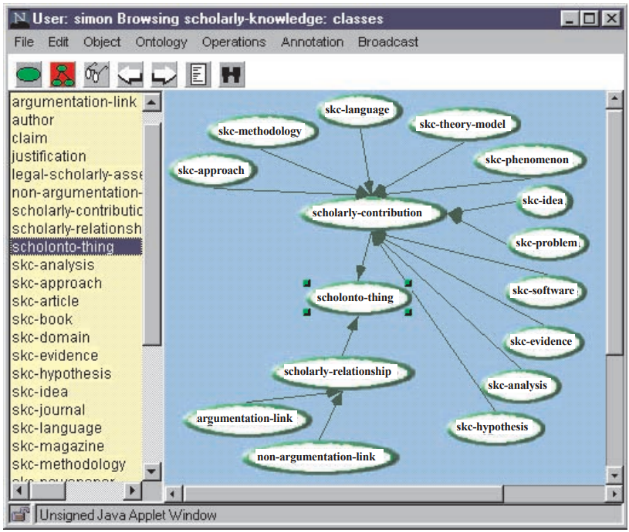
1. **ScholOnto:**

<https://link.springer.com/content/pdf/10.1007%2Fs007990000034.pdf>

במאמר המצורף בו מתואר שרת דיגיטלי המשמש כספרייה של מאמרים אקדמיים. השרת תומך באינטרפרטציה שונה של כל קורא למאמרים ובדו-שיח בנוגע לנושאים המאמרים, מה שמאפשר לחוקרים לתאר ולהתעמת דרך רשת סמנטית בה מתוארים התרומות וההקשרים של מאמר מסוים לשאר המאמרים בנושא.

במאמר מתוארים שירותי החישוב ששרת המבוסס על תיאוריית ה-ontology וממשקים אלטרנטיביים התומכים באינטראקציה ברשתות סמנטיות גדולות.

ע"מ להוסיף מאמר אל הספרייה על המפרסם להוסיף גם Claim על המאמר שלו. לכל claim יש "בעלים" שהוא agent (אדם או תוכנה), ישנה הצדקה כלשהי לקיום אותו claim ובעזרת claims ניתן ליצור יחסים גומלין עם claims אחרים או בין קונספטים שונים.

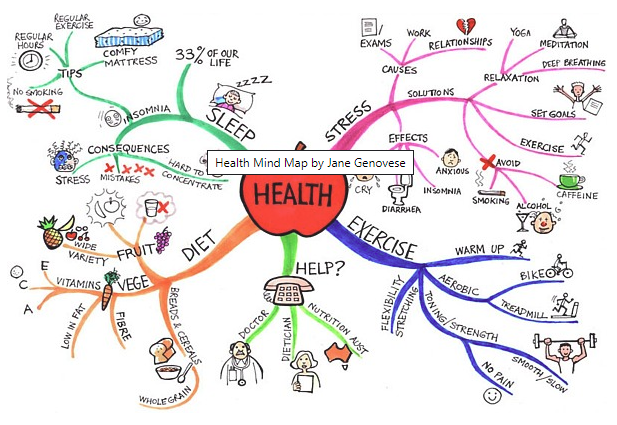
מבנה של Claim ע"פ המאמר: מבנה של מחלקות ב- ScholOnto ontology:

ההבדלים בין ScholOnto ontology למערכת שלנו המבוססת על מפות ME:

|  |  |
| --- | --- |
| **ScholOnto ontology** | **ME-Maps** |
| אלמנט יחיד שמכיל בתוכו את כל המידע המתאר את הנושא המוצג – ע"מ להבין יותר על הנושא יש לקרוא את המידע המצורף | ישנם כמה סוגי אלמנטים המציגים את המידע על הנושא בצורה יותר נגישה – task המציג את הנושא הרחב ו- qualities המאפיינים את הנושא. |
| מסובך – ע"מ להוסיף מאמר לרשת יש לנתח אותו ולכתוב claim המתאר אותו | ניתן לערוך |
| אין אפשרות לחיפוש |  |
|  |  |
|  |  |

1. **Mind maps:**

דיאגרמה לארגון מידע בצורה וויזואלית. הצגה היררכית של קשרים בין רכיבים המרכבים שלם מסוים, לרוב עוסקות בנושא אחד המופיע במרכז המפה. רעיונות משמעותיים מחוברים ישירות לרעיון המרכזי ורעיונות נוספים מחוברים אליהם.



|  |  |
| --- | --- |
| Mind maps | ME maps |
| המפה משמשת להרחבת הידע על נושא מסוים | המפה משמשת למציאת פתרון לבעיה מסוימת |
| כלי ויזואלי בלבד, אין אפשרות לקישור למקורות ידע נוספים | כל אלמנט במפה ניתן לקישור למאמרים/ דפי מידע נוספים |
| בכל מפה מוצג רעיון אחד מרכזי ואליו מקושרים רעיונות משנה | כל task מהווה מטרה/משימה שיש לבצע ובכך יכולים להיות כמה tasks במפה המתארים מטרות שונות |
| אין מודל ועל כן כל משתמש יוכל ליצור מפה שונה כרצונו | ישנו מודל מוגדר ופורמלי |

סקירת טכנולוגיות רלוונטיות:

* עבור הbackend נבחר בNodeJS – פלטפורמה להרצת קוד JavaScript מחוץ לדפדפן, קוד פתוח, מאפשרת התמודדות עם מספר רב של מחוברים במקביל. השפה מבוססת JS מה שמאפשר עבודה בשפה זהה גם ב- front end.

השפה פופולארית ובכך ישנן המון חבילות תוכנה בהן נוכל להשתמש.

אפשרויות נוספות שעמדו בפנינו הן Java, וAsp.net.

Java היא שפה מוכרת ומבוססת, אך פיתוח מערכות web באמצעותה הוא מסורבל לעומת האלטרנטיבות, שכן היא דורשת קומפילציה. בנוסף, ניהול משימות אסינכרוניות בjava דורש מימוש של observer ו-observable, בעוד בNode.JS המנגנון ממומש כבר בשפה באמצעות Promises. בהקשרים האלו, Asp.net דומה מאוד, ודורשת קומפילציה אף היא. גם מנגנון התהליכים האסינכרוניים דומה לJava.

אם כן, בחרנו עבור הbackend את nodeJS לאור התמיכה שלה בקוד אסינכרוני (שהוא חלק אינהרנטי מהשימוש בDB), ונוחות הפיתוח ללא צורך בקומפילציה. ניתן להריץ Live Server שמתעדכן בקוד החדש בכל רגע באופן אוטומטי.

* Front-End: Angular –היא framework של JavaScript, מבוססת TypeScript. מאפשרת שינויים ב- DOM בצורה דינאמית בעת הריצה. מקלה את הפיתוח לעומת JS. Angular היא שפה שמפותחת, נתמכת, וגם מפתחים באמצעותה ב-Google, מה שמבטיח תמיכה והמשך פיתוח בעתיד הנראה לעין. בנוסף, Angular מגיעה out of the box עם מימושים ופתרונות לבעיות שניתקל בהן, כמו router, form validation, httpClient ועוד.

אפשרויות נוספות שעומדות בפנינו הן Vue, jQuery ו-React. ארבע האפשרויות הן למעשה frameworks מבוססי JavaScript.

React היא למעשה סיפרייה שנועדה לפיתוח UI ע"י Facebook. ככזאת, היא אינה מגיעה עם כלל הכלים הנדרשים שמגיעים עם Angular, ותדרוש מאיתנו שימוש בספריות נוספות, מה שייצור תלות בחבילות חיצוניות ונקודות כשל אפשריות במקרה של שדרוגים או עדכונים בחבילות אלה. מבחינת ביצועים, React עדיפה על פני Angular, אך האפליקציה שלנו אינה דורשת ביצועים מהירים בצד הלקוח, אשר יממש תהליך ארוכים באמצעות GUI, ולא חישובים ארוכים ואוטומטיים.

לעומת React, Angular תומכת בdependency Injection, ומאלצת את המפתחים לחלוקה ברורה של MVC. שתי עובדות אלה מאלצות את הפיתוח להיות מודולרי ופשוט לבדיקה.

Vue.Js היא framework צעיר יחסית, ללא חברה גדולה שעומדת מאחוריו, או קהילת פיתוח גדולה וחזקה כמו שיש לReact ולAngular. לכן ישנן כרגע פחות חבילות מוכנות, ופחות תמיכה.

jQuery היא סיפרייה של JavaScript אשר מאפשרת לבצע מניפולצייה על הDOM באופן דינאמי בזמן הריצה.

בדומה לReact, שימוש בjQuery ידרוש מאיתנו שימוש בחבילות נוספות, שכן מדובר בחבילה בלבד, ולא בframework שלם. כמו כן, אין את ההפרדה המוגדרת לMVC. בנוסף, Angular תומכת בdata binding, בעוד בjQuery יש צורך לטפל בכל מקום שבוא נרצה לקשר בין data לבין תצוגה באופן ידני.

אם כן, בחרנו בAngular לאור העובדה שמדובר בframework עם תמיכה גדולה, שכולל פונקציונאליות רבה שנזדקק לה. כמו כן, החלוקה המוגדרת לMVC תאפשר לנו לכתוב קוד בצורה יעילה יותר.

* DBs: בנינו שני מבנים אפשריים לבסיס הנתונים ובימים אלו העברנו את המבנים והדרישות לדוקטורנטית במחלקה שחוקרת התאמות של DB לדרישות הטכניות של פרויקטים. מחכים לתשובה.

בין האפשרויות שעומדות בפנינו, ניתך למנות את השרתים הבאים: MySQL, MariaDB, PostgreSQL, SQLite, MongoDB, Neo4j, Cassandra.

SQLite היא מערכת קטנה, ומתאימה מאוד להטמעה בIoT ומערכות embedded. המערכת לא תומכת בחיבור של מספר משתמשים עם הרשאות שונות, וצוואר הבקבוק שלה הוא הכתיבה, אשר מתבצעת באופן סדרתי.

MySQL וMariaDB הן מערכת עם ארכיטקטורת שרת-לקוח, וmultithreaded. למעשה, MariaDB היא פיצול מתוך MySQL, לפני שזו נרכשה ע"י Oracle. בתוך הscope של הפרוייקט שלנו, ניתן לומר שאין הבדל מהותי בין שני הפתרונות.

ההבדל העיקרי בין השרתים שנסקרו עד כה לבין PostgreSQL, הוא שPostgreSQL מבוסס מידע אובייקט-רלציות, בעוד השרתים האחרים מחזיקים מידע כרלציות בלבד. מאפיין זה אינו מהותי במיוחד לפרוייקט שלנו, שאינו ממודל כאוסף של אובייקטים. מבחינת ביצועים, PostgreSQL מציג ביצועים דומים לMySQL וMariaDB. ישנה קהילה קטנה יותר של מפתחים סביב שרת זה לעומת האחרים.