מה"ט
המכון הממשלתי
להכשרה טכנולוגית

-ספר הפרויקט-

אינטגרציה
חיבור מהבסיס

מוגש על ידי:

גילה וגשל

ת.ז. 213792328

במסגרת התנסות מעשית (פרקטיקום)

אינטגרציה
חיבור מהבסיס

03/07/22

לכבוד

מכללת מרכז חרדי סניף סמינר גור אשדוד

שלום רב,

הנדון: אישור העסקה

הרינו לאשר כי מר/גב' גילה וגשל ת.ז. 213792328

מועסקת במשרד/ במפעל/ בחברה החל מיום 26/6/22 בהיקף של משרה מלאה.

התפקיד הנוכחי הינו מפתחת וכולל אפיון, מחקר, פיתוח וביצוע בדיקות תוכנה.

אין לנו כל התנגדות לביצוע פרויקט הגמר בהיקף של לפחות 300 שעות במסגרת עבודתו,

לשם כך ילווה על ידי אדם שתפקידו ראש צוות פיתוח.

בכבוד רב,

גיא צוקרמן

מתוך הצעת הפרויקט:

- **נושא/תחום ההתנסות המעשית (פעילות הסטודנט במסגרת ההתנסות)**

בניית מערכת שתתן ממשק נוח ויעיל לפקודות mongo dump & restore, משרת לשרת עפ"י בחירת הלקוח.
כיום, שחזור חלקי מסדי-נתונים מתבצע אך ורק בפקודת ה- **mongorestore command** המשתמש בנתונים המקומיים בלבד או המועברים ידנית לתיקיית ה-dumps במערכת. (dump ג"ב מתבצע ע"י commandה.)
אופציה זו דורשת משאבי התקנה וכן רלוונטית כברירת מחדל לכלל נתוני מסד-הנתונים המקומי בלבד ללא אפשרות בחירת נתונים חכמה.
המערכת תאפשר ביצוע פעולות dump & restore מכל ולכל מסד-נתונים פעיל ברשת הכללית והמקומית תוך מתן בחירת נתונים בצורה ייחודית ומסונכרנת ואפשרויות סינון מתקדמות ואוטומטיות.

- **פירוט הדרישות (פרוט המשימות הנדרשות מהסטודנט מתוך נהל פרויקט בתעשייה – פרקטיקום של המגמה הרלוונטית)**

משימות כלליות

✓ קליטה והיכרות - הכרת מרכז הלוינות במערך, תפקודו ותפקידיו. פגישת היכרות עם ראש הצוות והדרג המנהלי, קבלת עמדת עבודה וכלי פיתוח לצורך הביצוע. (מחשב, התקנות, חיבורים וכדו')

✓ אפיון ודרישות - קבלת מסמך דרישות לקוח פנים-ארגוני, אפיון, תכנון וחלוקת מבצוע.
הטמעת הארכיטקטורה ומתודולוגיית ניהול הפרויקט בשיטה האג'ילית כולל פגישת מעקב יומית, התמקדות בשיפור יכולת הצוות לספק תוצרים במהירות ולהגיב לדרישות העולות תוך כדי הפיתוח, לימוד עצמי וחקירת חומרים רלוונטיים.

פיתוח

✓ סביבה, טכנולוגיות ושפות - פיתוח בסביבת VScode, בשפות: React, node.js. שליפת ו/או השמת אובייקטים בבסיס הנתונים: mongoDB תוך חקירה ושימוש בספריית mongoTools לענפיה.

✓ כתיבת קוד - דגש על כתיבה חכמה, מתומצתת ויעילה בהתאם למקובלות clean-code.

✓ דרישות פונקציונליות למשתמש -
- ממשק ידידותי לבחירת סביבות רצויות עבור ביצוע פעולות dump & restore.
- סנכרון בין פעולות אוטומטיות שניתן להגדירן.
- סינון חכם ומגוון על מסדי הנתונים.
- ניהול וארגון קבצי קונפיגורציות.

Devops

✓ ספק יישומים - יצירת docker container המאפשר הרצת קוד מהירה ללא תלות בהתקנות.

בדיקות QA

✓ סביבת פיתוח - בדיקות בjest-framework, תוספת mock במידת הצורך לייעול תפקודי פונקציות.

✓ ביצוע - הרצת טסטים על גבי שרתי mongo של docker. ניטור הפעילות ע"י כתיבת bugs לlogger.

_____	_____	_____
חתימת הסטודנט	חתימת המנחה מהתעשייה	חתימת המנחה מהמכללה

- **הערות ראש המגמה במכללה בה לומד הסטודנט**

_____

- **אישור ראש המגמה**

שם: _____ חתימה: _____ תאריך: _____

אינטגרציה
חיבור מהבסיס

מבוא

משרד הביטחון אחראי לביטחון המדינה ברמה המדינית, הצבאית והאזרחית. מטפל בסוגיות הבטחון באמצעות הצבת מענה מערכתי לבניין צה"ל ועוצמתו, תוך כדי חיזוק יכולת ההרתעה שלו.

יחידה 9900 מאגדת מספר יחידות בצה"ל, כחלק מאגף המודיעין. משימתה היא איסוף מודיעין חזותי לשימוש צה"ל ושאר כוחות הביטחון.

המערך מורכב מאנשי טכנולוגיה ואנשי מודיעין המשלבים את הידע במטרה לבנות כלים רלוונטיים בעידן המידע לאמ"ן ולכוחות בשטח.

המערך מאגד בתוכו את **מרכז הלווינות** (היחידה הוקמה בשנת 1997 ומשמשת כסוכנות לווייני הביון הלאומית של מדינת ישראל), יחידת המיפוי, מנהלת לוחמ"ם (לוחמה מועשרת מודיעין), יחידת הרחפנים, מרכז המיצוי והגאו-אינט, ענף דיגיטל ועוד.

הפרויקט מתייחס למערך הלווינות ונועד לתת מענה לצרכי פיתוח של אנשי היחידה.

כיום, ישנן פעולות רבות הנדרשות לצורך הפיתוח ומתבצעות באופן ידני בשורת הפקודה, הדבר גורם לסרבול רב, לבזבוז משאבי אנוש וזמן יקר.

הפרויקט הנוכחי ייתן מענה לענין בכך שיספק שירותי mongo dump & restore אוטומטיים באמצעות ממשק משתמש יעיל וידידותי.

המערכת תאפשר ביצוע פעולות dump & restore מכל ולכל מסד-נתונים פעיל ברשת הכללית והמקומית תוך מתן בחירת נתונים בצורה ייחודית ומסונכרנת ואפשרויות סינון מתקדמות ואוטומטיות.

מטרת הספר הינה הצגת הפרויקט מהיבטיו השונים, התרשמות מהנעשה בו ע"מ להפיק ממנו את המירב.

הספר מכיל מדריך למתכנת המחולק לעקרונות התכנון ועקרונות הבניה.

עקרונות התכנון כוללים תיאור פרויקט מבחינה תכנותית ואלגוריתמית, תהליכי המערכת, תיאור בסיסי הנתונים והבעיות שנלוו לפיתוח התוכנה בשילוב דרכי הפתרון.

עקרונות הבניה מתארים את המחלקות והפונקציות העיקריות הקיימות בפרויקט בצרוף חלקי קוד רלוונטיים.

בנוסף, הספר מכיל מדריך למשתמש, כלל צילומי מסך ופירוט מגוון האפשרויות הכלולות במערכת.

אני תקווה כי תמצאו פרויקט זה יעיל ומועיל ותשאבו השראה לחידושים נוספים.

ובנימה אישית...

שבועות של השקעה וטיפוח חושים:

ראיה מערכתית, אפיון וירידה לתתי פרטים.

קשב רב והתנהלות רב-צוותית, נטילת אחריות בלי פשרות,

להריח חדשנות ולהתפתח בכלי טכנולוגיית-על

חשה בקצות האצבעות את הדגדוג שקורא לי לחקור ולא להפסיק ללמוד

נותרתי בטעם טוב...

אינטגרציה
חיבור מהבסיס

תוכן עניינים:

1. הגדרות, דרישות ותיאור כללי

1.1 תיאור כללי
1.2 תיאור חומרה
1.3 תיאור תוכנת המערכת
1.4 תיאור פונקציות המערכת

2. ממשקים חיצוניים

3. ממשק אדם מכונה

3.1 כללי
3.2 תיאור מסכים

4. ארגון קבצים ומבנה נתונים

4.1 ארגון קבצים
4.2 מבנה נתונים

5. תכנון

5.1 כללי
5.2 עקרונות התכנות
5.3 תיאור אלגוריתמים
5.4 בדיקות המערכת

6. תוצרי הפרויקט

7. ביבליוגרפיה

1. הגדרות, דרישות ותיאור כללי

1.1 תיאור כללי

בעידן הטכנולוגי בו אנו חיים, התנהלות חיי היום-יום מבוססת על מערכות ממוחשבות רבות, התפתחויות טכנולוגיות מובילות לשינויים רבים ומעצימות את חשיבות יכולת הפיתוח, כאשר מצד אחד עומדת החשיבות של רמת הפיתוח וביצועי הקוד ומצד שני במקום לא פחות חשוב חווית המשתמש ונוחות.
ניתן לומר כי הדבר הראשון העומד בראש מעיניו של מתכנת הוא לפתח תוכנה שתקל על המשתמש, ותהווה עבורו פתרון נוח לשימוש.
בפרויקט זה חקרתי היטב את הנושא, הגדרתי את הדרישות העומדות בפני, ובניתי בהתאם מערכת שתתן ממשק נוח לפקודות mongo dump & restore משרת אחד לשרת אחר.

1.2 תיאור חומרה

סביבת פיתוח:
חומרה: i7 32GB RAM
מערכת הפעלה: Windows10
מסד נתונים: MongoDB

עמדת משתמש מינימאלית
חומרה: i3 4GB RAM
מערכת הפעלה: Windows7
חיבור לאינטרנט: נדרש

1.3 תיאור תוכנת המערכת

<u>כללי</u>

- Server - Node.js v14.18.1 (express).
  גישה למסד הנתונים באמצעות MongoClient & MongoTools .
- Client - React lib המשמשת לבניית אלמנטים אינטראקטיביים באתרי אינטרנט, שימוש בתבניות עיצוב מיוחדות של material-UI.
  שפות: JS ו TypeScript, HTML ו-CSS.
- מסד נתונים - MongoDB v^4.7.0

- בדיקות QA - בדיקות ב jest-framework תוספת mock במידת הצורך לייעול תפקודי פונקציות.
- Devops - ספק יישומים, יצירת docker container המאפשר הרצת קוד מהירה ללא תלות בהתקנות.

<u>כלי תוכנה לפיתוח המערכת</u>

- Microsoft VS code
- MongoDB-Compass
- Docker
- Postman
- GitHub Desktop

1.4 תיאור פונקציות המערכת

המערכת מכילה מספר פונקציות עיקריות:

- Dump
- Restore
- Filter
- Auto jobs
- Async Dump (by queue)
- Edit configurations
- Logger

Dump & Restore – הפונקציות הדומיננטיות ביותר:

<u>Dump</u> - העתקת מידע נבחר מסביבה* רצויה בקובץ דחוס לתיקיה מוגדרת.

למשתמש יוצגו שמות הסביבות הקיימות, (מיובאות מרשימת הסביבות המוגדרת בקובץ קונפיגורציה) על המשתמש לבחור סביבה רצויה.

במקרה ולא תבחר סביבה – תבחר הסביבה שהוגדרה מראש כברירת מחדל.

עבור הסביבה הנבחרת יוצגו שמות הDBS. (המערכת תפנה לשרת הmongo ותייבא משם את שמות הDBS, ייתכנו מספר DBS, המשתמש יוכל לבחור DBS כרצונו)

במקרה ולא ייבחרו DBS - יתבצע Dump לכל הDBS.

עבור כל DB נבחר יוצגו שמות הCollections התואמים. (המערכת תפנה לשרת הmongo ותייבא משם את שמות הCollections, ייתכנו מספר Collections, המשתמש יוכל לבחור Collections כרצונו)

אפשרות זו תוצג במידה והמשתמש בחר לפחות אחד מן הDBS.

במקרה ולא ייבחרו Collections, הפונקציה תבצע Dump לכל הCollections.

תוספים:

- מחיקת dumps ישנים מתיקיית ה- dumps של המערכת.
- Restore מקובץ dump קיים.

*סביבות- כינוי לקבוצת שרתים הפועלים יחד, בין שרתים אלו נמצא את שרתי הmongo שלנו.

Restore - שחזור מידע קיים מתיקיית Dumps במערכת לסביבה רצויה.
למשתמש יוצגו שמות הסביבות הקיימות, (מיובאות מרשימת הסביבות המוגדרת בקובץ קונפיגורציה) על המשתמש לבחור סביבה רצויה.
במקרה ולא תבחר סביבה – יתבצע Dump ופונקציית Restore לא תופעל.

פירוט פונקציות נוספות:

חיווי הצלחה/כשלון - לאחר שהמשתמש יפעיל את הבקשות באמצעות לחצן SUBMIT הקורא לפונקציות, תופיע על המסך הודעת חיווי כדלהלן:
במידה והפעולה הצליחה- הודעה מאשררת.
במידה והפעולה נכשלה- הודעת שגיאה בתוספת אפשרות לצפות בתוכן השגיאה.
במקרה ופעולת הdump לא צלחה- פעולת הrestore לא תתבצע.

Filter - משתמש שיודע לכתוב שאילתה לסינון נתונים יוכל להכניס שאילתה מורכבת ויחודית המחזירה לUI collection מסונן היטב עליו ניתן לבצע Dump.

Auto jobs - אפשרות לrestore & dump אוטומטיים עפ"י הגדרות שבחר המשתמש מראש כגון: תאריך, שעה, חזרה בכל יום/ שבוע/ חודש וכדו'.
הפונקציה מתבצעת באמצעות תזמוני cron job.

Async Dump - תור לביצועי dumps שנרשמו למערכת. מטרת התור הינה מניעת גישה מקבילית לDatabase לצורך הגנה על נתוני הסביבה.

Edit configurations - אפשרות ניהול קבצי קונפיגורציות דרך הUI – מחיקת, עריכת והוספת סביבות רבות בו זמנית.

Logger - כתיבת כל שלב ביצועי במערכת בקובץ מיועד במחשב המשתמש כך שכאשר יפנה המשתמש לתמיכה טכנית בעקבות שגיאות במערכת, יוכל כל המתמצא במערכת להבין היכן ארעה שגיאה.
הפונקציה מתבצעת באמצעות pino-logger לcontainer.

2. ממשקים חיצוניים

Docker's MongoDB server - על מנת לוודא ביצועי קוד מקסימליים, הרצתי מספר שרתי mongo וירטואליים ע"ג Docker-desktop.
תוספת זו אפשרה מעקב חוזר ונשנה אחר מספר סביבות במקביל, סנכרון מושלם בין חלקי השרתים וייעול מבצוע.

Git - מיזוג חלקי הפרויקט בשלבים למטרת שמירה על הסדר הטוב ואחסון יעיל. משימות הפרויקט התבצעו בעבודת צוות כך שכל שותף יצר אזור אישי ושלח PullRequest בסיום כל חלק משמעותי למיזוג הקוד הפועל בענף המערכת הכללי.(master)
ההתחברות נערכה באמצעות אפליקציית GitHub.com למתכנת.

Trello - לוח משימות מקוון לצורך חלוקה עבודה אפקטיבית, מעקב פרטני אחר כל משימה, אפשרות קביעת Deadline לטווח רחב ושליחת התראות בעת צורך.
בנוסף, בכרטיס המשימה ניתן לתקשר בין השותפים בזמן אמת לבירור ולווידוא יעדים.

3. ממשק אדם מכונה

3.1 כללי

ממשק משתמש נמצא בלב חוויית המשתמש: זהו המקום שבו נוצרת האינטראקציה בין האדם (המשתמש) לבין המכונה.
בנייה ועיצוב של ממשק משתמש הנו תהליך עבודה אשר דורש מחשבה ומלאכה רבה, אמנם בסופו המשתמשים יוכלו ליהנות מיכולת התמצאות נוחה באתר, ניווט נוח וידידותי ועוד.
בבניית המערכת התמקדתי בעיצוב GUI אינטואיטיבי לחוויית משתמש מושלמת באמצעות אלמנטים ויזואליים וטקסטואליים:

- התאמה רספונסיבית לגודל המסך.
- הודעות מערכת קריאות ומפורטות.
- עיצוב בהיר ונעים לעין ועוד.

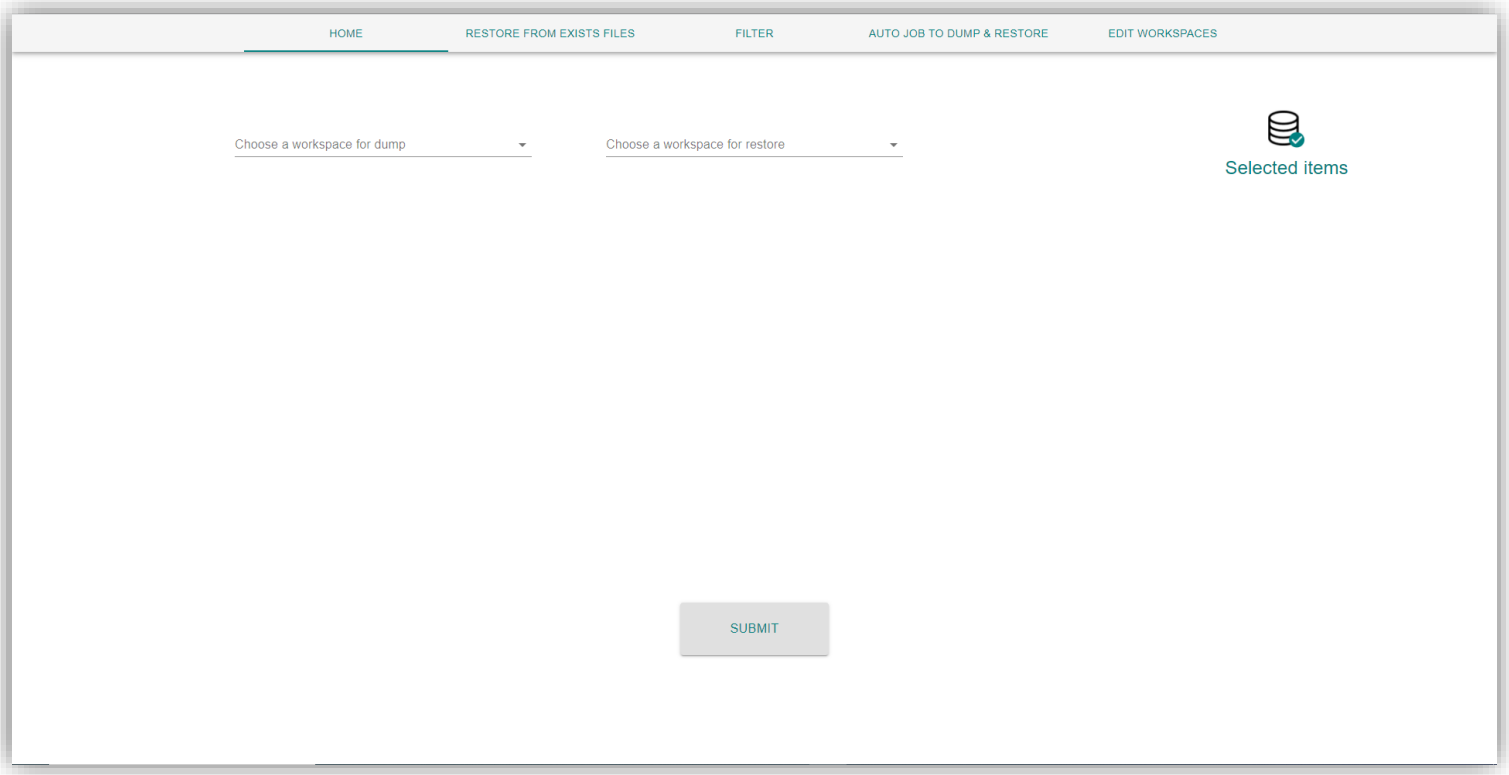
אינטגרציה
חיבור מהבסיס

3.2 תיאור המסכים

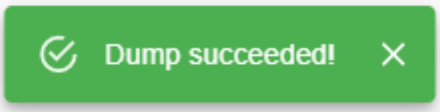
להלן צילומי מסכים מובילים במערכת:

<u>מסך ראשי</u>

- בחירת סביבות
- לחצן "בצע"

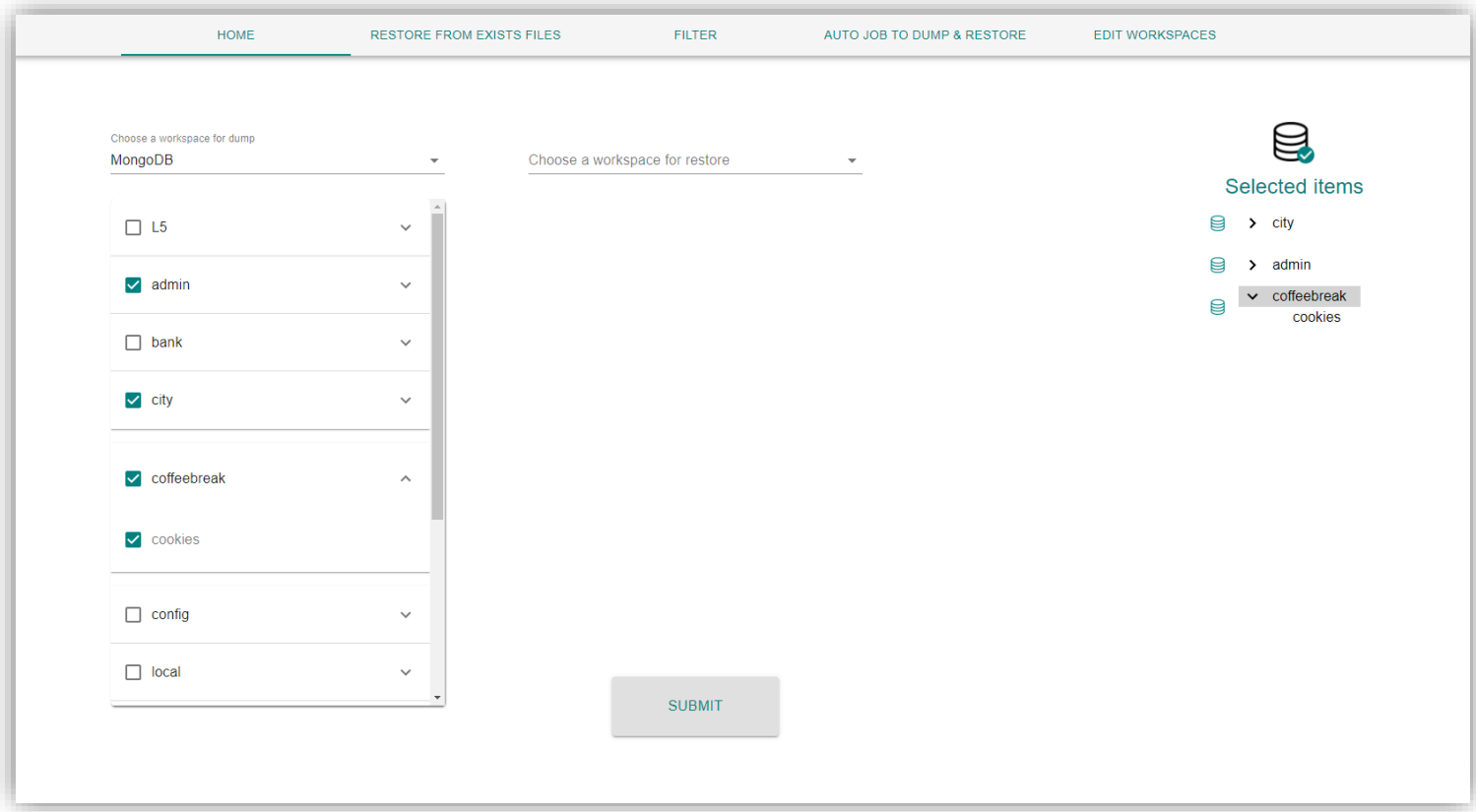


- חיווי הצלחה

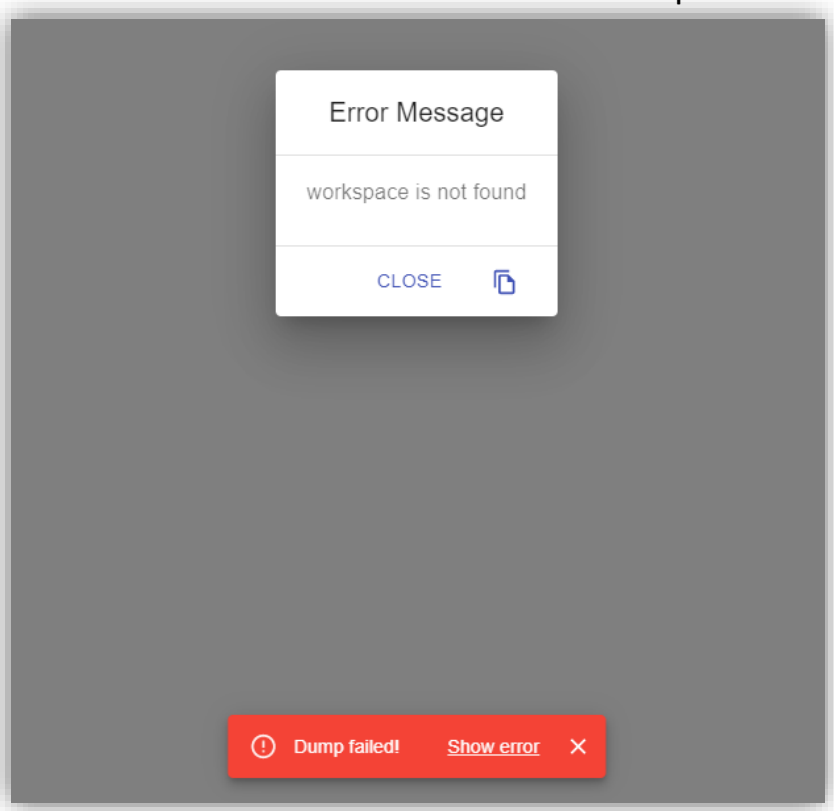


אינטגרציה
חיבור מהבסיס

- בחירה מרובה



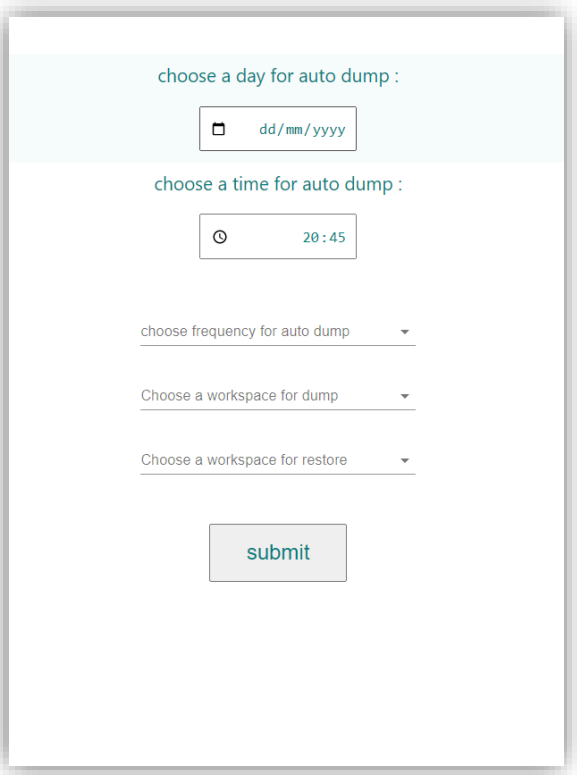
- חיווי כשלון ואפשור צפייה בשגיאה



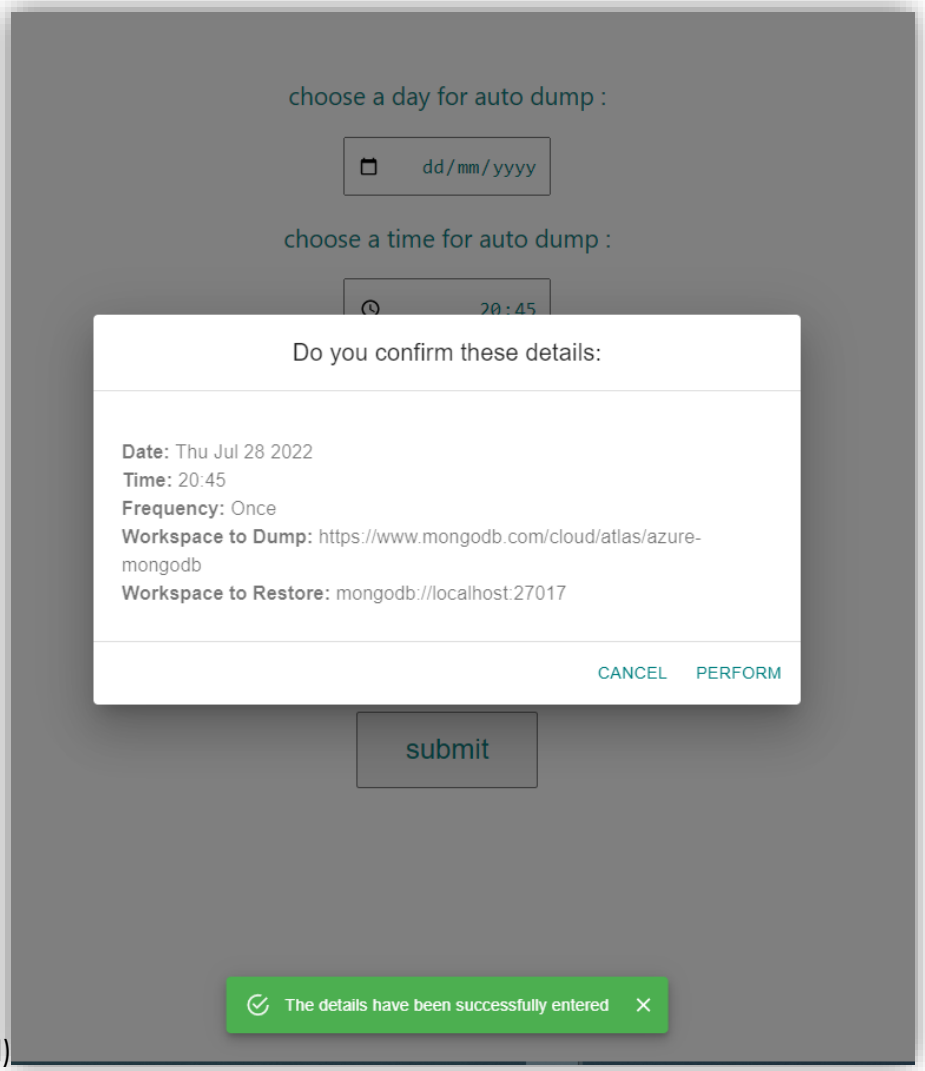
אינטגרציה
חיבור מהבסיס

Auto job

- הזנת פרטי תזמונים אוטומטיים



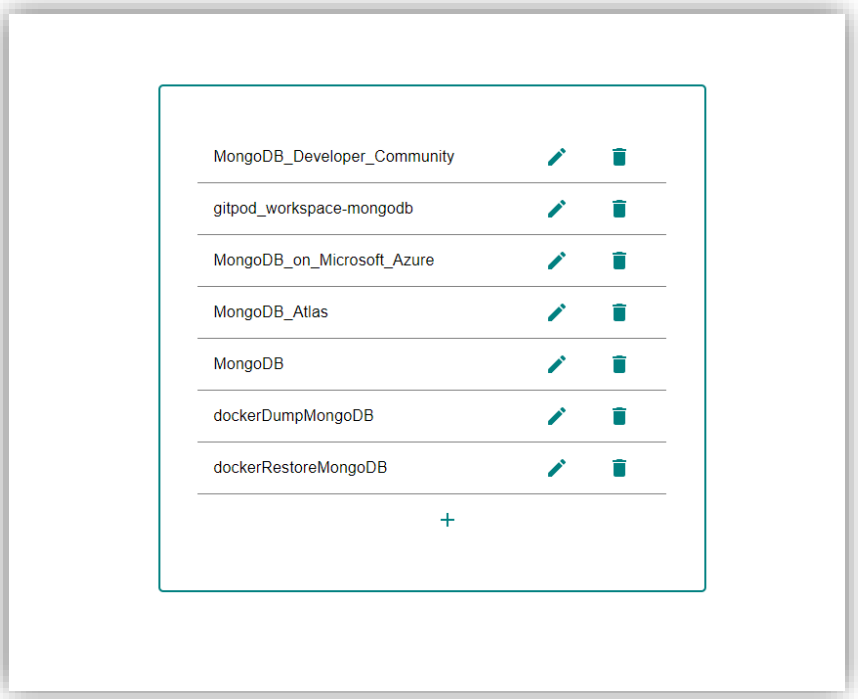
- אשרור פרטים



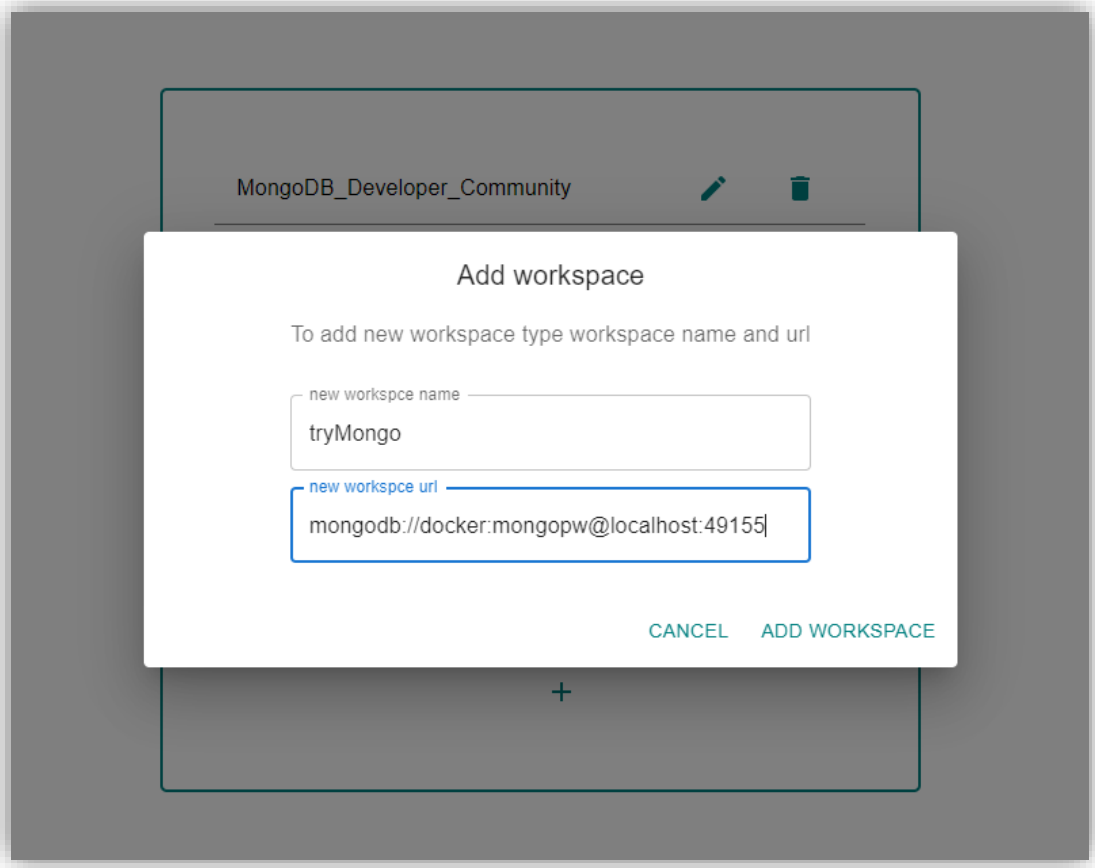
אינטגרציה
חיבור מהבסיס

Edit configuration

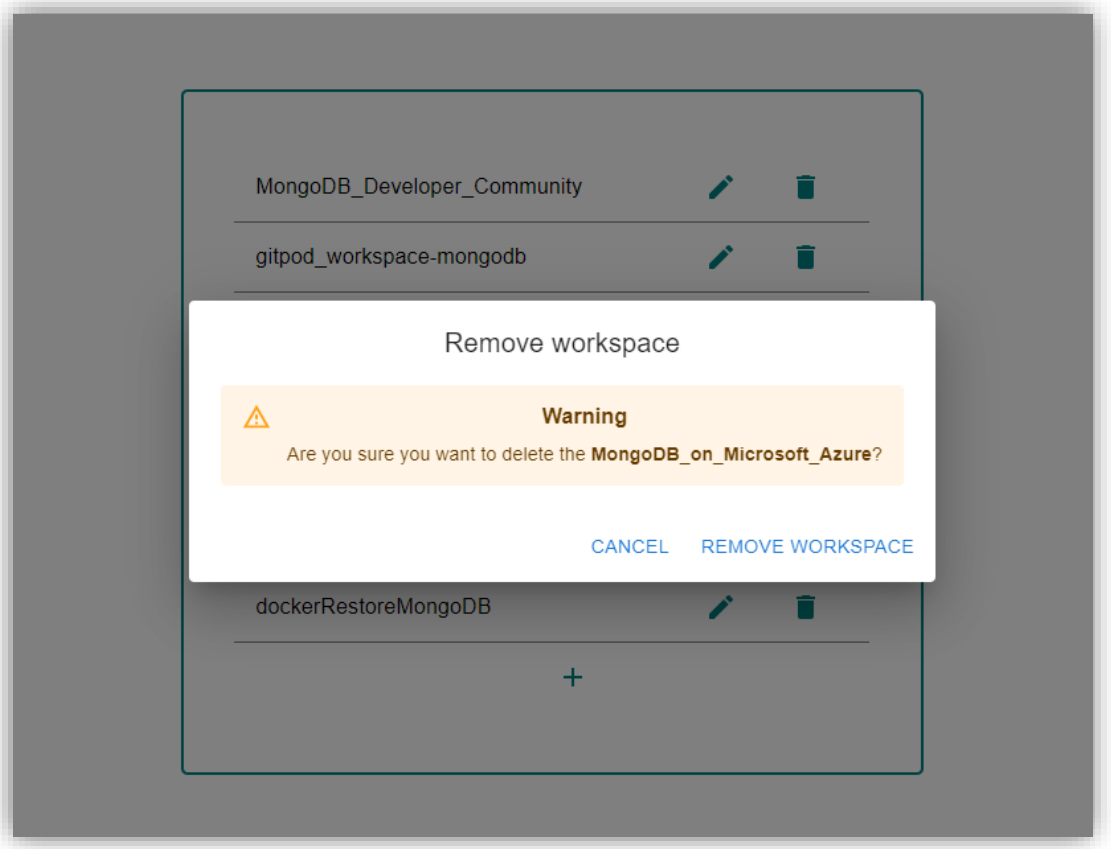
- הצגת הסביבות הקיימות



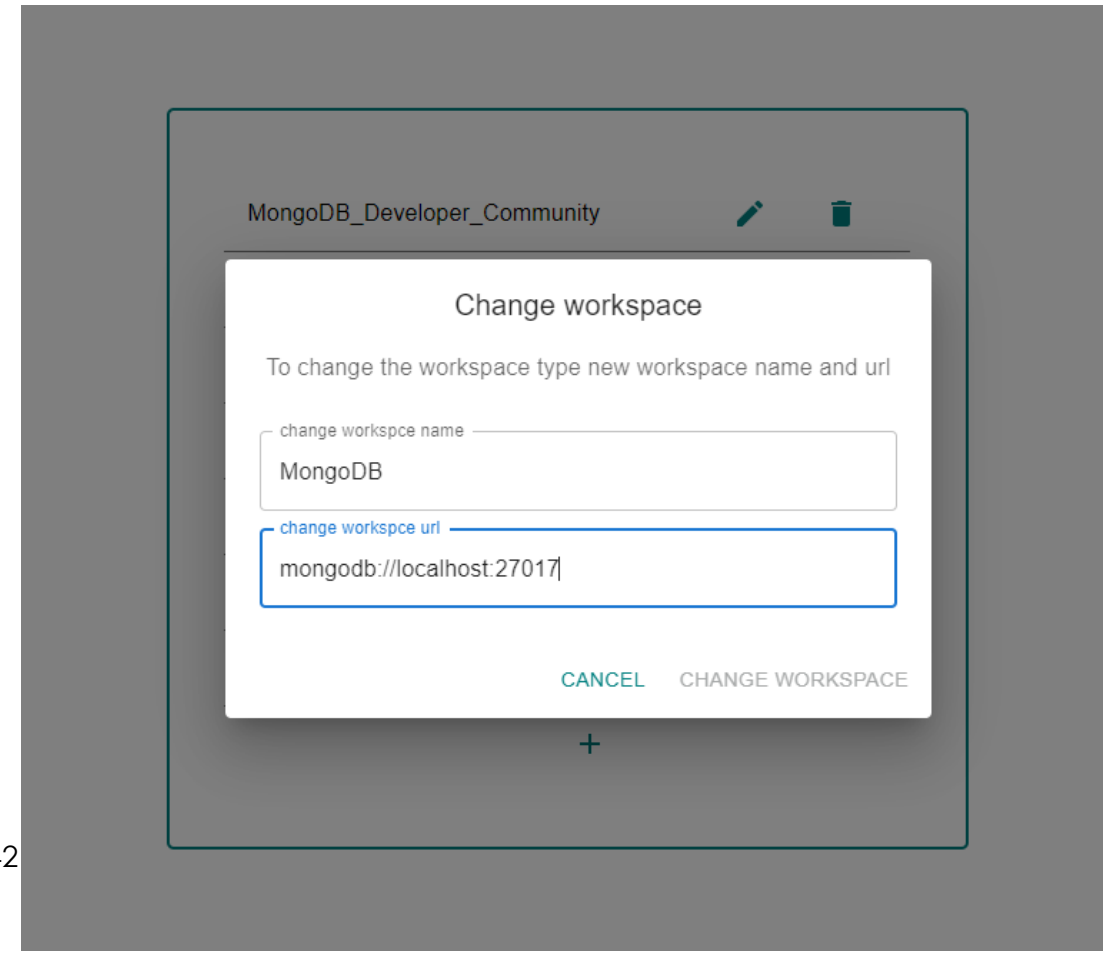
- הוספת סביבה חדשה



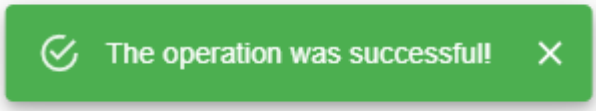
- מחיקת סביבה קיימת לאחר וידוא נוסף



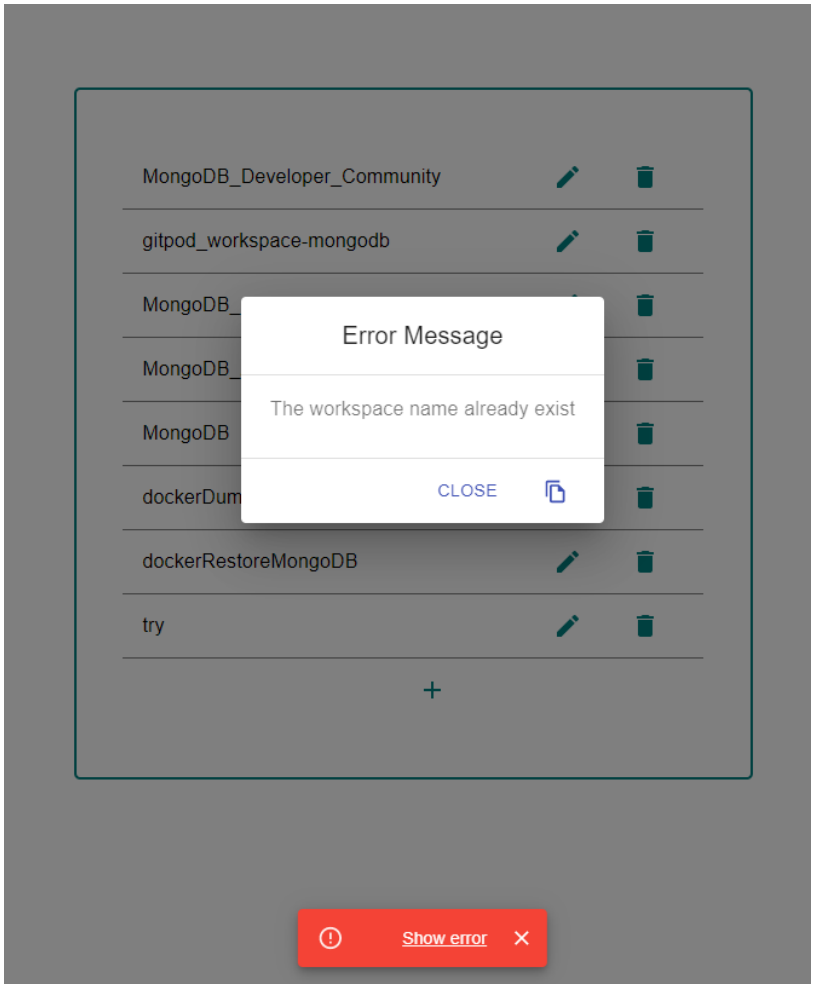
- עריכת פרטי סביבה קיימת



- חיווי הצלחה



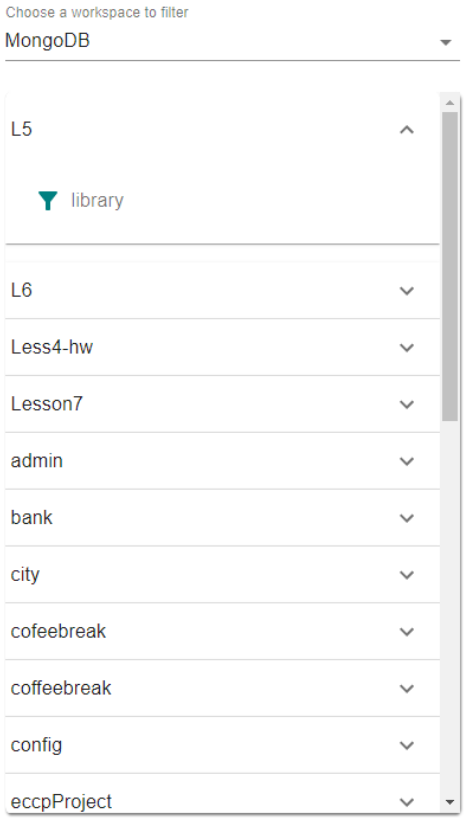
- חיווי כשלון ואפשור צפייה בשגיאה



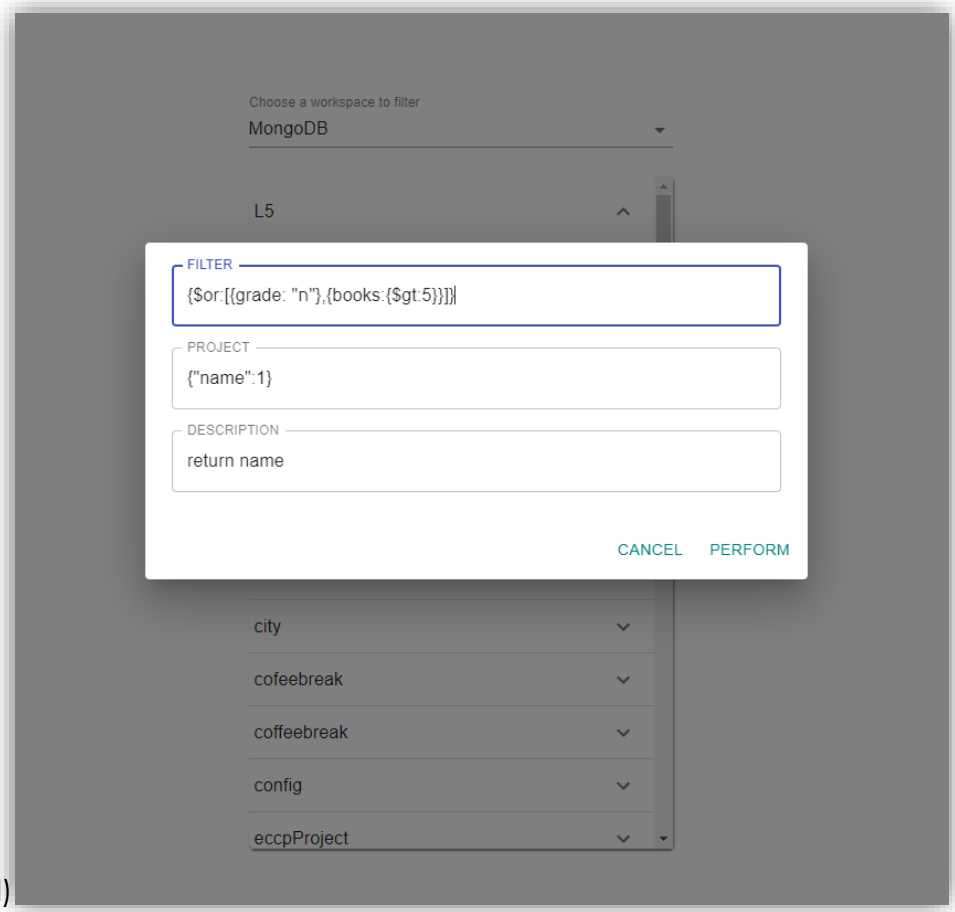
אינטגרציה
חיבור מהבסיס

Filter

- בחירת collection לסינון

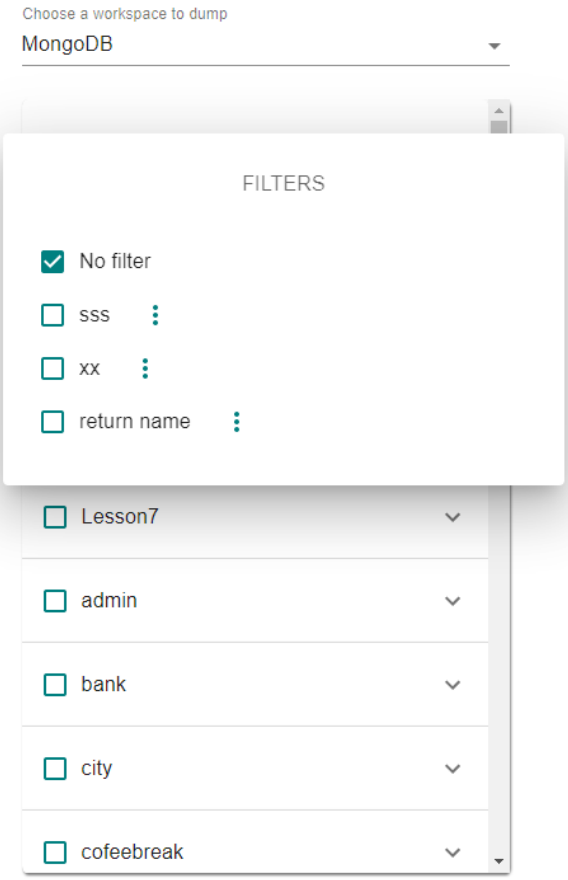


- דיאלוג אפשור סינון נתונים



אינטגרציה
חיבור מהבסיס

- הצגת תוצאות הסינון

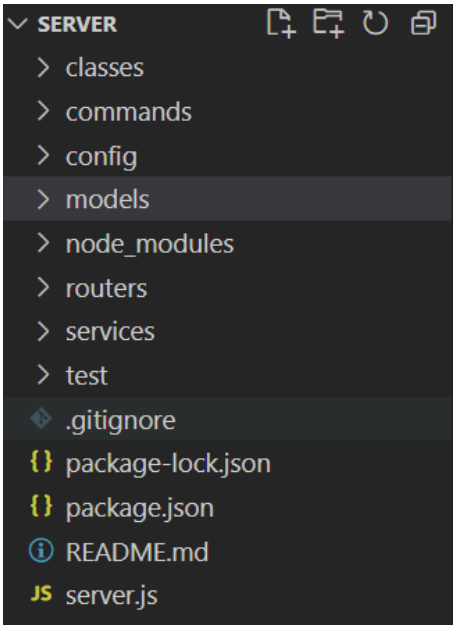


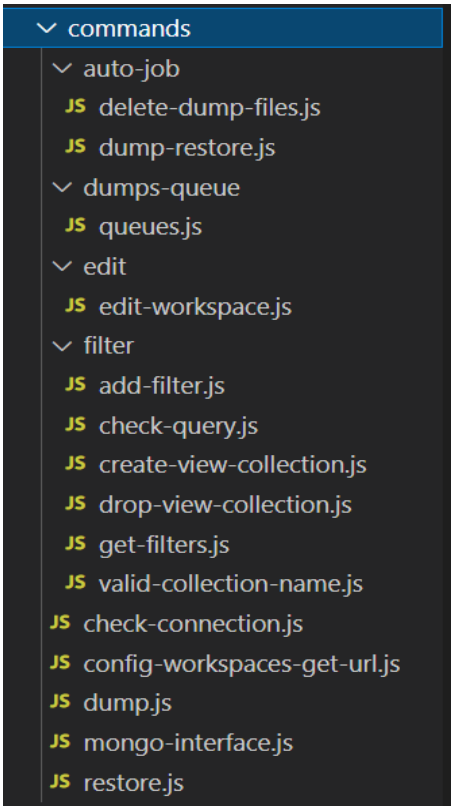
4. ארגון קבצים ומבנה נתונים

4.1 ארגון קבצים

- <u>צד שרת</u>

צד השרת נבנה בטכנולוגיית Node,בשפת JavaScript והוא מכיל כמה תיקיות וקבצים:





תיקיית commands - המכילה את הפונקציות הכלליות הנדרשות לביצוע הפעולות העיקריות במערכת.

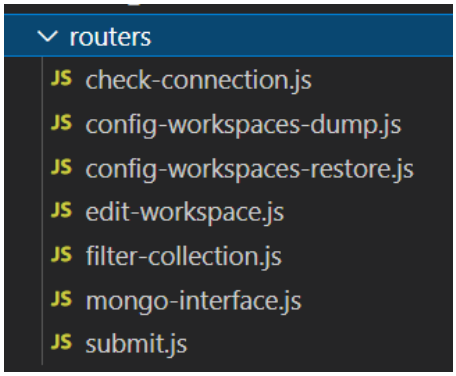
תיקיית config - המכילה קבצי קונפיגורציה, נקרא גם קובץ תצורה, קובץ המכיל פרמטרים להתאמה מקומית של תוכנה ולהכנסת דוגמה תהליכית.

תיקיית models – מכילה את הclasses השונים, כל class מייצג מבנה אובייקט כלשהוא, בהתאם למבני הנתונים השונים.

תיקיית routers – בה נמצאים הrouters של הפרויקט, כל route אחראי על מספר פונקציות המתייחסות לנושא משותף וניתן לגשת אליהם באמצעות API.

תיקיית test – מבדקי המערכת בשעת הפיתוח, התעדוף התבצע ע"י כיסוי כל תרחישי הבדיקות לפי מתודולוגיות בדיקה ,תוכנות, לוגים ועוד.
בדיקות תוכנה הן תהליך שנועד להעריך את איכותה של תוכנה ועמידתה בדרישות שהוצבו לה ומהוות חלק אינטגרלי מתהליכי הנדסת תוכנה והבטחת איכות תוכנה.

מספר קבצים ותיקיות נוספים בעלי אחריות על תפקוד התוכנה הכללי בצד השרת, בהם נמצאת גם תיקיית node_modules – המשמשת מטמון עבור המודלים החיצוניים, npm שהפרויקט תלוי בם, התיקיה אינה קיימת בgit משום שמומלץ למפתחים להתקינה בכל פעם מתחילה.



קובץ server.js הכולל את ייבוא הroutersים
והפקודות הכלליות שתומכות במערכת.
ניתן לצפות בהיררכיית התיקיות והקבצים של צד השרת <-

- <u>צד לקוח</u>

רכיבי צד הלקוח (=האפליקציה) נבנו בארכיטקטורת react.
כל רכיב במערכת (=component) מכיל שילוב שפתי של css – עיצוב נראות באמצעות useStyle, HTML – אחראי על המבנה הויזואלי וגריד מסך וקוד TypeScript – הכולל לוגיקת-על (בהתבסס על שפת JS) ובדיקות.
תיקיית context מכילה reducer (=הקשר) המאפשר להעביר מידע לכל האפליקציה מבלי להשתמש באלמנטים נוספים.
ניתן לצפות בהיררכיית התיקיות והקבצים של צד הלקוח <-

4.2 מבני נתונים

אופן שמירת הנתונים הגנריים מתבצע באמצעות קבצי JSON.

- dump details-שמירת נתוני הdump בקובץ מקומי.


```
[{
        "date": "2022-07-28T15:23:47.768Z",
        "workspaceName": "MongoDB",
        "workspaceURL": "mongodb://localhost:27017",
        "dbs": [{
                "db": "admin",
                "collections": [
                    "gili",
                    "Ahuvi"
                ]
            },
            {
                "db": "config",
                "collections": []
            },
            {
                "db": "ooo",
                "collections": [
                    "clients"
                ]
            },
            {
                "db": "tryRestore",
                "collections": [
                    "products",
                    "suppliers",
                    "clients"
                ]
            }
        ],
        "isSucceeded": {
            "isSucceeded": true,
            "dumpFile": "MongoDB_2022-07-28T15.23.47.768Z"
        }
    },
```

דוגמאות לתוצרים:

- רשימת סביבות קיימות במערכת לביצועי הפעולות

```
{
    "MongoDB_Developer_Community": "https://www.mongodb.com/community/forums/t/design-for-multi-customer-solution/129820",
    "gitpod_workspace-mongodb": "https://hub.docker.com/r/gitpod/workspace-mongodb",
    "MongoDB_on_Microsoft_Azure": "https://www.mongodb.com/mongodb-on-azure",
    "MongoDB_Atlas": "https://www.mongodb.com/cloud/atlas/azure-mongodb",
    "MongoDB": "mongodb://localhost:27017",
    "dockerDumpMongoDB": "mongodb://docker:mongopw@localhost:49153",
    "dockerRestoreMongoDB": "mongodb://docker:mongopw@localhost:49154"
}
```

- Environment - נתונים גלובליים במערכת

```
{
    "APP_PORT": 3131,
    "DUMP_FILE": "D:/משתמשים/project/Documents/GitHub/Dump",
    "DEFAULT_DUMP_URL": "mongodb://localhost:27017"
}
```

- הודעת חיווי כלליות הניתנות לגישה משותפת לכלל הפקודות

```
{
    "ERR_1": "The workspace not found",
    "ERR_2":"The file not found",
    "ERR_3" : "The workspace url already exist",
    "ERR_4":  "The workspace name already exist",
    "ERR_5":  "Failed to add",
    "ERR_6":  "Failed to remove",
    "ERR_7":  "Failed to change"
}
```

אינטגרציה
חיבור מהבסיס

שמירת נתוני הפילטור במסד נתונים mongoDB.

filters.query_data

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

DOCUMENTS N/A | TOTAL SIZE N/A | AVG. SIZE N/A | INDEXES

FILTER | OPTIONS

ADD DATA | VIEW | Displaying documents 1 - 4

query_data

	_id ObjectId	workspace String	db String	collection String	query String	projection String	description String
1	62e237c43e2783e01d4d0bbb	"mongodb://localhost:27017"	"L5"	"library"	"{"$or":[{"grade": "π"},{"books	"{}"	"sss"
2	62e23c9b3b65d139ecc75a1c	"mongodb://localhost:27017"	"L5"	"library"	"{"$or":[{"grade": "π"},{"books	"{}"	"xx"
3	62e2449bc5388f723fc8fbf2	"mongodb://localhost:27017"	"L6"	"update"	"{}"	"{}"	"ccc"
4	62e2d17cfd16ed4b261386c6	"mongodb://localhost:27017"	"L5"	"library"	"{"$or":[{"grade": "π"},{"books	"{"name":1}"	"return name"

filters.query_data

Documents | Aggregations | Schema | Explain Plan

FILTER

ADD DATA | VIEW | {}


```
{
    "_id": {
        "$oid": "62e237c43e2783e01d4d0bbb"
    },
    "workspace": "mongodb://localhost:27017",
    "db": "L5",
    "collection": "library",
    "query": "{\"$or\":[{\"grade\": \"π\"},{\"books\":{\"$gt\":5}}]}",
    "projection": "{}",
    "description": "sss"
}

{
    "_id": {
        "$oid": "62e23c9b3b65d139ecc75a1c"
    },
    "workspace": "mongodb://localhost:27017",
    "db": "L5",
    "collection": "library",
    "query": "{\"$or\":[{\"grade\": \"π\"},{\"books\":{\"$gt\":5}}]}",
    "projection": "{}",
    "description": "xx"
}

{
    "_id": {
        "$oid": "62e2449bc5388f723fc8fbf2"
    },
    "workspace": "mongodb://localhost:27017",
    "db": "L6",
    "collection": "update",
    "query": "{}",
    "projection": "{}",
    "description": "ccc"
}

{
    "_id": {
        "$oid": "62e2d17cfd16ed4b261386c6"
    },
    "workspace": "mongodb://localhost:27017",
    "db": "L5",
    "collection": "library",
    "query": "{\"$or\":[{\"grade\": \"π\"},{\"books\":{\"$gt\":5}}]}",
    "projection": "{\"name\":1}",
    "description": "return name"
}
```

5. תכנון

5.1 כללי

האתר נבנה במודל שרת-לקוח

<u>צד השרת:</u>
צד השרת נבנה בטכנולוגיית Node,בשפת JavaScript.
<u>צד הלקוח:</u>
רכיבי צד הלקוח (=האפליקציה) נבנו בארכיטקטורת react.
כל רכיב במערכת (=component) מכיל שילוב שפתי של css – עיצוב נראות באמצעות useStyle ובאמצעות תבניות עיצוב יחודיות (templates) של materialUI, HTML – אחראי על המבנה הויזואלי וגריד מסך וקוד TypeScript – הכולל לוגיקת-על (בהתבסס על שפת JS) ובדיקות.

5.2 עקרונות התכנות

מטרתי בפרוייקט זה הייתה לכתוב קוד קצר, יעיל וממוקד, הבנוי מחלקי קוד קטנים, שימוש מקסימלי בטכנולוגיות העומדות לרשותי ובכלי התוכנה שנבחרו לצרכי הפיתוח. הפרויקט נבנה תוך חלוקה נכונה בין ממשקי המשתמש, תפקוד לוגי של המערכת והתקשרות למערכות אחרות.

5.3 תיאור אלגוריתמים

להלן מספר קטעי קוד עיקריים:

- שרת – WebApi

<u>Dump</u>
פונקציית ה-dump מקבלת מהלקוח אובייקט על פיו עליה לפעול.
עד פעולת ה-dump עצמה, הפונקציה בונה לעצמה אובייקט מלא בהתאם לצורך.
כלומר: אם הפונקציה קיבלה אובייקט עם שם סביבה בלבד, הפונקציה ממלאת לעצמה מערך DBS בהתאם לשם הסביבה.

```
const response = await importDBS({ workspace: dataForDump.workspace_to_dump })
dataForDump.dbs = response.map(_db => ({ db: _db, collections: [] }))
```

וכן, במקרה שהאובייקט על פיו הפונקציה פועלת מלא בשמות BDS ללא שמות Collections, הפונקציה מעדכנת שמות Collections פר- DB.

```
const response = await importCollections({ workspace: dataForDump.workspace_to_dump, db: _db.db })
```

כעת, כאשר האובייקט מעודכן ומלא, הפונקציה פונה לפעולת ה-dump עצמה, שפונה לתיקיה שם רוצה הלקוח לשמור את קבצי ה-dump, יוצרת בתוכה תיקיה נוספת ששמה שם הסביבה והתאריך הנוכחי, ובתוך תיקיה זו יוצרת את הקבצים עפ"י שם Collection_DB .

```
for (let col of _db.collections) {
    await mongoTools.mongodump({
        url: dataForDump.workspace_to_dump,
        path: `${config}/${nameFile}`,
        db: _db.db,
        collection: col,
        fileName: `${_db.db}.gz`
    });
}
```

Filter

להלן פונקציה המחזירה חיווי על תקינות שאילתת הפילטור.

commands > filter > JS check-query.js > checkQuery

```
const { MongoClient } = require('mongodb');
const { insertFilter } = require('./add-filter')

async function checkQuery(data) {

    try {
        const url = `${data.workspace}/`;
        const client = await MongoClient.connect(url)
        const connect = client.db(data.db);
        await connect.collection(data.collection)
            .find(JSON.parse(data.query), JSON.parse(data.projection))
            .toArray();
        client.close()
        const object = await insertFilter(data)
        return { filterSucceeded: true, object: object }
    } catch (error) {
        return { filterSucceeded: false, error: error.message }
    }
}

module.exports = { checkQuery };
```

להלן פונקציה השומרת את אובייקט השאילתה ב-DB.

```
const { MongoClient } = require('mongodb');
const config = require('../../config/env.json');

async function insertFilter(data) {

    try {
        const url = `${config.FILTERS_PORT}/`;
        const client = await MongoClient.connect(url)
        const connect = client.db(config.FILTER_DATA);
        const object = await connect
            .collection(config.FILTER_COLLECTION)
            .insertOne(data)
        client.close()
        return object
    } catch (error) {
        throw error
    }

}

module.exports = { insertFilter };
```

להלן פונקציה המחזירה את כל המידע על הפילטורים הקיימים במערכת.

commands > filter > JS get-filters.js > getFilters > filters

```
1   const { MongoClient } = require('mongodb');
2   const config = require('../../config/env.json');
3
4
5   async function getFilters() {
6
7       try {
8           const url = `${config.FILTERS_PORT}/`;
9           const client = await MongoClient.connect(url)
10          const connect = client.db(config.FILTER_DATA);
11          const filters = await connect
12              .collection(config.FILTER_COLLECTION)
13              .find({}, {}).toArray()
14          client.close()
15          return filters
16      } catch (error) {
17          throw error
18      }
19  }
20
21  module.exports = { getFilters };
22
```

להלן פונקציה היוצרת Collection זמני מפולטר עליו יתבצע ה-Dump.

```
4   async function createViewCollection(data) {
5
6       try {
7           const url = `${data.workspace}/`;
8           let viewCollectionName = `${data.collection}_${validCollectionName(data.description)}`
9           const client = await MongoClient.connect(url)
10          const connect = client.db(data.db);
11          let pipeline = [
12              { '$match': JSON.parse(data.query) }
13          ]
14
15          if (data.projection !== '{}')
16              pipeline = [...pipeline, { '$project': JSON.parse(data.projection) }]
17
18              await connect.createCollection(
19              viewCollectionName,
20              {
21                  "viewOn": data.collection,
22                  "pipeline": pipeline
23              }
24          )
25          client.close()
26          return viewCollectionName
27      }
28      catch (error) {
29          return error
30      }
31
32  }
33
34  module.exports = { createViewCollection };
```

להלן פונקציה ההופכת את שם ה-Collection לשם תקין בלי תווי זבל.

```
commands > filter > JS valid-collection-name.js > validCollectionName
1   function validCollectionName(name) {
2       return name.replace(/^[^\w]+|[^\w]+$/g, "_")
3       .replace(/ /g,'_')
4   }
5
6   module.exports = { validCollectionName };
7
```

להלן פונקציה המוחקת את ה-Collection לאחר שהתבצע עליו Dump.

```
commands > filter > JS drop-view-collection.js > dropViewCollection
const { MongoClient } = require('mongodb');

async function dropViewCollection(data) {

    try {
        const url = `${data.workspace}/`;
        const client = await MongoClient.connect(url)
        const connect = client.db(data.db);
        await connect.dropCollection(data.collection)
        client.close()
        return "Collection deleted"
    }
    catch (error) {
        return error;
    }

}

module.exports = { dropViewCollection };
```

להלן חלק מ-Component המרנדרת חלונית להכנסת השאילתה.

```
return (
  <>
    <IconButton color="primary" aria-label="upload picture" component="label" onClick={handleClickOpen}>
      <FilterAltIcon className="overrideColor" />
    </IconButton>
    <Dialog open={open} onClose={() => handleClose("Cancel")} aria-labelledby="form-dialog-title">
      <DialogContent>
        <TextField
          label="FILTER"
          variant="outlined"
          fullWidth
          style={{ marginBottom: '2vh' }}
          onChange={(event) => handleTextInputChange("filter", event)}
        />
        <TextField
          id="project"
          label="PROJECT"
          variant="outlined"
          fullWidth
          style={{ marginBottom: '2vh' }}
          onChange={(event) => handleTextInputChange("project", event)}
        />
        <TextField
          id="description"
          label="DESCRIPTION"
          variant="outlined"
          fullWidth
          style={{ marginBottom: '2vh' }}
          onChange={(event) => handleTextInputChange("description", event)} />
        {
```

להלן פונקציה ההופכת את הקלט שהוכנס עבור השאילתה לאובייקט.

src > components > filter > validQuery.js > validCollectionName

```
export const validCollectionName = (query) => {
    try {
        JSON.parse(query)
    }
    catch {
        let i = 0
        while (query.indexOf("{", i) !== -1) {
            i = query.indexOf("{", i)
            query = query.substring(0, i + 1) + '"' + query.substring(i + 1, query.length);
            i += 3
        }
        i = 0
        while (query.indexOf(":", i) != -1) {
            i = query.indexOf(":", i)
            query = query.substring(0, i) + '"' + query.substring(i, query.length);
            i += 3
        }
    }
    return query
}
```

להלן Component השולחת את השאילתה לבדיקה בשרת ומרנדרת הודעת חיווי בהתאם.

```
async function fetchData() {
    try {
        let response = await axios.get(`${config.API_URL}/filter_collection/check_query`, {
            params: data
        })
        setIsFilterSucceeded(response.data)
    } catch (error) {
        console.error('Error:', error);
    };
}

useEffect(() => {
    fetchData();
}, [])

const handleClose = () => {
    setOpen(false)
}

return (
    <>
        {isFilterSucceeded ?
            <Snackbar open={open} autoHideDuration={3000} onClose={handleClose}>
                <Alert
                    onClose={handleClose}
                    severity={isFilterSucceeded.filterSucceeded ? "success" : "error"}>
                             
                    <span style={{ cursor: 'pointer' }}>
                        {isFilterSucceeded.filterSucceeded ?
                            " The filter was executed successfully" :
                            isFilterSucceeded.error
                        }
                    </span>
                </Alert>
            </Snackbar>
            : <></>
        }
    </>
);
```

להלן Component המציגה למשתמש את השאילתות הקימות במערכת + אפשרות

```
const isNone = _ => {
    checked.map(item => item.checked = false)
}

const ifNone = _ => {
    let none = true
    checked.map(f => { if (f.checked) none = false })
    return none
}

const isChecked = f => {
    checked.find(item => item.id == f).checked = !checked.find(item => item.id == f).checked
}

const ifChecked = f => {
    return checked.find(item => item.id == f).checked
}

const toShowQuery = f => {
    showQuery.find(item => item.id === f).show = !showQuery.find(item => item.id === f).show;
};

const ifShowQuery = (f) => {
    return showQuery.find(item => item.id === f).show;
}

const handleClick = state => {
    setOpen(state);
}
```

Edit Workspace

בניית הפונקציות התבצעה בעזרת יצירת class עבור יעילות הקוד

```
module.exports = class EditWorkspace {

    constructor(data) {
        this.name = data.workspaceName;
        this.url = data.workspaceUrl;
        this.workspaceConfig = JSON.parse(fs.readFileSync(`config/workspaces-configuration.json`, 'utf8'));
        this.isSucceeded = { isSucceeded: true };
    }
```

פונקציה להוספת סביבה חדשה לקובץ קונפיגורציה

```
async addWorkspace() {
    try {
        Object.entries(this.workspaceConfig).filter(([key, value]) => {
            if (value === this.url) {
                this.isSucceeded = { isSucceeded: true, errorMsg: errorMessage.ERR_3 };
            }
            if (key === this.name) {
                this.isSucceeded = { isSucceeded: false, errorMsg: errorMessage.ERR_4 };
            }
        });
    } catch {
        this.isSucceeded = { isSucceeded: false, errorMsg: errorMessage.ERR_1 }
    }
    if (this.isSucceeded.isSucceeded) {
        try {
            this.workspaceConfig[this.name] = this.url;
            fs.writeFileSync(`config/workspaces-configuration.json`, JSON.stringify(this.workspaceConfig), 'utf8');
            if (!this.isSucceeded.errorMsg) {
                this.isSucceeded = { isSucceeded: true };
            }
        } catch (error) {
            this.isSucceeded = { isSucceeded: false, errorMsg: errorMessage.ERR_5 };
        }
    }
    return this.isSucceeded;
}
```

פונקציה למחיקת סביבה חדשה לקובץ קונפיגורציה

```
async removeWorkspace() {
    if (this.workspaceConfig[this.name] && this.workspaceConfig[this.name] === this.url) {
        try {
            delete this.workspaceConfig[this.name];
            fs.writeFileSync(`config/workspaces-configuration.json`, JSON.stringify(this.workspaceConfig), 'utf8');
            this.isSucceeded = { isSucceeded: true }
        } catch {
            this.isSucceeded = { isSucceeded: false, errorMsg: errorMessage.ERR_6 };
        }
    }
    else {
        this.isSucceeded = { isSucceeded: false, errorMsg: errorMessage.ERR_1 };
    }
    return this.isSucceeded;
}
```

פונקציה לשינוי סביבה קיימת בקובץ קונפיגורציה

```
async changeWorkspace(newData) {
    const newName = newData.newWorkspaceName;
    const newUrl = newData.newWorkspaceUrl;
    const saveName = this.name;
    const saveUrl = this.url;

    try {
        if (newUrl !== this.url && newName === this.name) {
            this.workspaceConfig[this.name] = newUrl;
            fs.writeFileSync(`config/workspaces-configuration.json`, JSON.stringify(this.workspaceConfig), 'utf8');
            this.isSucceeded = { isSucceeded: true };
        }
        else {
            if (newName)
                this.name = newName;
            if (newUrl)
                this.url = newUrl;
            let responseAdd = await this.addWorkspace();
            if (responseAdd.isSucceeded) {
                this.name = saveName;
                this.url = saveUrl;
                let responeRemove = await this.removeWorkspace();
                if (responeRemove.isSucceeded)
                    this.isSucceeded = { isSucceeded: true };
                else {
                    this.name = newName;
                    this.url = newUrl;
                    responeRemove = await this.removeWorkspace();
                    this.isSucceeded = { isSucceeded: false, errorMsg: errorMessage.ERR_7 };
                }
            }
            else
                this.isSucceeded = { isSucceeded: false, errorMsg: errorMessage.ERR_7 };
        }
    } catch {
        this.isSucceeded = { isSucceeded: false, errorMsg: errorMessage.ERR_7 };
```

```
return (
    <>

        <Tooltip title="Delete">
            <Button className="overrideColor" onClick={handleClickOpen}>
                <DeleteIcon />
            </Button>
        </Tooltip>
        {props.workspaceChoose.workspaceName && props.workspaceChoose.workspaceUrl ?
            <Dialog open={open} onClose={handleClose}>
                <DialogTitle>Remove workspace</DialogTitle>
                <DialogContent>
                    <Alert severity="warning">
                        <AlertTitle><strong>Warning</strong></AlertTitle>
                        Are you sure you want to delete the <strong>{props.workspaceChoose.workspaceName}</strong>?
                    </Alert>
                </DialogContent>
                <DialogActions>
                    <Button onClick={handleClose}>CANCEL</Button>
                    <Button onClick={handleRemove} >REMOVE WORKSPACE</Button>
                </DialogActions>
            </Dialog>
            : <></>
        }
    </>
);
```

```
return (
    <span >
        <Tooltip title="Add new">
            <Button className={`${classes.EditButton} hover overrideColor`} onClick={handleClickOpen}>
                <AddIcon />
            </Button>
        </Tooltip>
        <Dialog open={open} onClose={handleClose}>
            <DialogTitle>Add workspace</DialogTitle>
            <DialogContent >
                <DialogContentText className={classes.DialogContentText}>
                    To add new workspace type workspace name and url
                </DialogContentText>
                <TextField onChange={(event) => handleChange(event, "name")}
                    autoFocus
                    margin="dense"
                    className={classes.TextField}
                    label="new workspce name"
                    variant="outlined" />
                <TextField
                    InputProps={{
                        classes: { notchedOutline: classes.specialOutline }
                    }}
                    autoFocus
                    margin="dense"
                    className={classes.TextField}
                    onChange={(event) => handleChange(event, "url")}
                    label="new workspce url"
                    variant="outlined" />
            </DialogContent>
            <DialogActions >
                <Button onClick={handleClose} className="overrideColor">CANCEL</Button>
                <Button onClick={handleAdd} disabled={canAdd} className="overrideColor">ADD WORKSPACE</Button>
            </DialogActions>
        </Dialog>
    </span>
);
```

Auto job

להלן הפונקציה autoJobDumpAndRestore אשר מקבלת אובייקט המכיל :

תאריך וזמן להפעלת ה – dump בצורה אוטומטית הפונקציה מחזירה האם הנתונים נקלטו ובהתאם לזאת יוצג למשתמש הודעת חיווי תואמת .

הפונקציה מפעילה בתוכה פונקציה נוספת שעושה את ה – dump ואם קימת בחירת סביבה ל restore את ה – restore ורושמת לקובץ Log הצלחה או כישלון .

```
6   async function autoJobDumpAndRestore(objectToDumpAuto) {
7
8       try {
9           const date = new Date(objectToDumpAuto.date);
10          const month = date.getMonth() + 1;
11          const day = date.getDate();
12          const hour = date.getHours();
13          const minute = date.getMinutes();
14
15          let timeToAutoDump;
16
17          if (objectToDumpAuto.frequency == 'Every Hour')
18              timeToAutoDump = '0 * * * *';
19
20          if (objectToDumpAuto.frequency == 'Everyday')
21              timeToAutoDump = `${minute} ${hour} * * *`;
22
23          if (objectToDumpAuto.frequency.includes('Weekly')) {
24              let dayInWeek = objectToDumpAuto.frequency.split(':');
25              timeToAutoDump = `${minute} ${hour} * * ${dayInWeek[1]}`;
26          }
27
28          if (objectToDumpAuto.frequency == 'Monthy')
29              timeToAutoDump = `${minute} ${hour} ${day} * *`;
30
31          if (objectToDumpAuto.frequency == 'Yearly' || objectToDumpAuto.frequer
32              timeToAutoDump = `${minute} ${hour} ${day} ${month} *`;
33
34          objectToDump = {
35              "workspace_to_dump": objectToDumpAuto.state.workspace_to_dump,
36              "dbs": objectToDumpAuto.state.dbs
37          }
38
39          const autoDump = cron.schedule(timeToAutoDump, async () => {
40              await doDumpAndRestore(objectToDump, objectToDumpAuto.state.worksp
41          });
42          autoDump.start();
```

הפונקציה מחזירה תיבת בחירה עם זמני התדירות להפעלת Auto Dump.

בבחירת זמינות שבועית יוצג למשתמש אפשרות לבחירת יום.

```
14  function Frequency(props) {
15
16      const classes = useStyles();
17
18      let defaultProps = {
19          options: [{}],
20          getOptionLabel: _ => _
21      };
22
23      const frequencyTimes = ['Once', 'Every Hour', 'Everyday', 'Weekly', 'Monthy', 'Yearly']
24
25      if (frequencyTimes) {
26          const options = Object.entries(frequencyTimes).map(([key, val]) => {
27              return ({ name: key, frequency: val });
28          });
29          defaultProps = {
30              options: options,
31              getOptionLabel: op => op.frequency
32          };
33      }
34      return (
35          <div className={classes.Autocomplete}>
36              <Autocomplete
37                  style={{ width: '20vw' }}
38                  {...defaultProps}
39                  id="select-Frequency"
40                  blurOnSelect
41                  onChange={async (event, value) => {
42                      if (value)
43                          props.handleClick(value.frequency)
44                  }}
45                  renderInput={(params) => <TextField {...params} label="choose frequency for auto dump "
46              />
47          </div>
48      );
49  }
50
51  export default Frequency
52
```

להלן פונקציה אשר מתבצעת בעת בקשת תדירות של שבוע ונותנת למשתמש את האפשרות לבחור באיזה יום בשבוע יתבצע ה - Auto Dump .

הפונקציה מחזירה תיבת בחירה עם ימי השבוע .

```
    const Days = ['Sunday', 'Monday', 'Third ', 'Wednesday', 'Thursday', 'Friday']


    let defaultProps = {
        options: [{}],
        getOptionLabel: _ => _
    };

    if (Days) {
        const options = Object.entries(Days).map(([key, val]) => {
            return ({ name: key, day: val });
        });
        defaultProps = {
            options: options,
            getOptionLabel: op => op.day
        };
    }
    return (
        <div className={classes.Autocomplete}>
            <Autocomplete
                style={{ width: '20vw' }}
                {...defaultProps}
                id="select-days"
                blurOnSelect
                onChange={async (event, value) => {
                    if (value) {
                        props.handleClick(value.day);
                    }

                }}
                renderInput={(params) => <TextField {...params} label="Choose a day in week " 
            />
        </div>
    );
}

export default Days;
```

קטע קוד מתוך פונקציית החיווי למשתמש .

הפונקציה פונה עם הנתונים שקיבלה ל - server להפעלת ה- dump האוטומטי.

```
const postData = async () => {
    const autojobsDetails = {
        date: new Date(props.date + " " + props.time),
        frequency: props.frequency,
        state: state
    }

    const requestOptions = {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(autojobsDetails),
    };
    try {

        const response = await fetch(`${config.API_URL}/dump_auto_job`, requestOptions)
        const answer = await response.json()
        setNotification(answer)
    } catch (error) {
        throw error;
    };
};

useEffect(() => {
    postData()
}, [])
useEffect(() => {
    if (Object.keys(notification).length !== 0) {
        notification.isSucceeded ?
            setNotification_show({ severity: "success", message: " The details have been successfully entered" })
            : setNotification_show({ severity: "error", message: " Error in receiving the details" })
    }
}, [notification])
```

Dump Queue

המחלקה:

```
const { isTheDatabaseUsed } = require('../commands/dumps-queue');
const { doDump } = require('../commands/dump');

class DumpQueue {
    constructor() {
        this.dump_queue = [];
    }

    async addItem(dump) {···
    }

    async removeItem(dump) {···
    }

    async emitNextDump(workspace) {···
    }

    async dump(dump) {···
    }

    async tryDump(dump) {···
    }
}
module.exports = DumpQueue;
```

אינטגרציה
חיבור מהבסיס

פונקציות פנימיות:

AddItem

```
async addItem(dump) {
    let isExistWorkspace = false;

    const canDump = await this.tryDump(dump);
    if (!canDump) {
        this.dump_queue.forEach((key) => {
            if (Object.keys(key)[0] === dump.dumpDetails.workspace_to_dump) {
                Object.values(key)[0].push(dump);
                isExistWorkspace = true;
            }
        });

        if (!isExistWorkspace) {
            this.dump_queue = [...this.dump_queue, {
                [dump.dumpDetails.workspace_to_dump]: [dump],
            }];
        }
    }

    dump.finish.on('remove', () => {
        this.dump_queue.removeItem(dump);
    });
}
```

removeItem

```
async removeItem(dump) {
    for (let i = 0; i < Object.values(this.dump_queue).length; i += 1) {
        if (Object.keys(Object.values(this.dump_queue)[i])[0] === dump.dumpDetails.workspace_to_dump) {
            if (Object.values(Object.values(this.dump_queue)[i])[0].length === 1) { this.dump_queue.splice(i, 1); } else {
                const removeDumpIndex = Object.values(Object.values(this.dump_queue)[i])[0].findIndex((removeDump) => JSON.str
                Object.values(Object.values(this.dump_queue)[i])[0].splice(removeDumpIndex, 1);
            }
        }
    }
}
```

Emit

```
async emitNextDump(workspace) {
    for (let i = 0; i < Object.values(this.dump_queue).length; i += 1) {
        if (Object.keys(Object.values(this.dump_queue)[i])[0] === workspace) {
            Object.values(Object.values(this.dump_queue)[i])[0].forEach((tryEmitNext) => {
                const canDump = this.tryDump(tryEmitNext);
                if (canDump) return;
            });
        }
    }
}
```

אינטגרציה
חיבור מהבסיס

Dump & tryDump

```
async dump(dump) {
    const dumpResult = await doDump(dump);
    dump.finish.emit('finish', dumpResult);
    this.removeItem(dump);
    this.emitNextDump(dump.dumpDetails.workspace_to_dump);
}

async tryDump(dump) {
    const cantDoDump = await isTheDatabaseUsed(dump.dumpDetails);
    return !cantDoDump ? this.dump(dump) : false;
}
```

Queue

```
const { importDBS } = require('./mongo-interface');

let usedDumpDatabases = [];

async function markUsedDatabase(dataForDump, date) {…
}

async function isTheDatabaseUsed(dataForDump) {…
}

async function DeleteUsedDatabaseAfterDump(date) {…
```

סימון DBS בשימוש

```
async function markUsedDatabase(dataForDump, date) {
  date = date.toISOString();
  const workspaceDetailsToSave = {};
  const dbsPerWorkspace = {};
  dbsPerWorkspace[dataForDump.workspace_to_dump] = [];
  workspaceDetailsToSave[date] = dbsPerWorkspace;

  if (dataForDump.dbs.length === 0) {
    try {
      const response = await importDBS({ workspace: dataForDump.workspace_to_dump });
      dataForDump.dbs = response.map((_db) => ({ db: _db, collections: [] }));
    } catch (error) {
      return { isSucceeded: false, errorMsg: 'workspace is not found' };
    }
  }

  workspaceDetailsToSave[Object.keys(workspaceDetailsToSave)[0]][dataForDump.workspace_to_dump] = dataForDump.dbs.map((el) => el.db);
  usedDumpDatabases = [...usedDumpDatabases, workspaceDetailsToSave];
}
```

בדיקה האם DB מסוים נמצא בשימוש כעת

```
async function isTheDatabaseUsed(dataForDump) {
  for (const dt of usedDumpDatabases) {
    if (Object.keys(Object.values(dt)[0])[0] === dataForDump.workspace_to_dump) {
      if (dataForDump.dbs.length === 0) { return true; }
      for (const db1 of dataForDump.dbs) {
        for (const db2 of Object.values(Object.values(dt)[0])[0]) {
          if (db1.db === db2) { return true; }
        }
      }
    }
  }
  return false;
}
```

מחיקת DB לאחר ביצוע Dump

```
async function DeleteUsedDatabaseAfterDump(date) {
  usedDumpDatabases = usedDumpDatabases.filter((dt) => {
    new Date(Object.keys(dt)[0]).toTimeString() !== date.toTimeString()
          || new Date(Object.keys(dt)[0]).toDateString() !== date.toDateString();
  });
}
```

5.4 בדיקות המערכת

בדיקות תקינות – ולידציה בצד client למניעת הזנת נתונים שגויים העלולים לגרום לקריסת המערכת.

Jest-framework – מסגרת בדיקה בקוד פתוח

7. תוצרי הפרויקט

- ניסיון בתכנון ואפיון עצמאי למערכת ממסד ועד.
- חקירת מגוון אפיקי פיתוח ע"מ להכיר ולבחור את המיטב.
- בנית מערכת תקשורתית API לסנכרון שרת-לקוח
- לימוד לעומק שפת react, התעסקות עם קומפוננטות מורכבות ויצירה דינמית של קומפוננטות.
- עבודת צוות לויאלית ומהוגנת באמצעות אפליקציית GitHub למתכנת.
- פיתוח פרואקטיביות, יכולת אלתור וחשיבה יצירתית, התמודדות עצמאית עם אתגרים.
- רכישת כלי פיתוח חדשניים באמצעות חקירות ובירור.
- פיתוח יכולת לימוד עצמי ברמה מקסימלית.

8. ביבליוגרפיה

http://en.wikipedia.org
https://stackoverflow.com
https://reactjs.org
https://v4.mui.com
www.tutorialsteacher.com
nodejs.org
www.mongodb.com