



שם תלמידה: יובל ברזלי

3.....	מבוא
3.....	תיאור תכולת הספר
3.....	הרקע לפרויקט
3.....	תהליך המחקר
4.....	אילו חידושים יש בפרויקט
5.....	בחירת הנושא ומוטיבציה לעבודה
5.....	על איזה צורך הפרויקט עונה? איזה פתרון הפרויקט הזה בא לתת?
5.....	הפתרון
5.....	בגרסאות הבאות
6.....	מדריך למשתמש
6.....	הוראות התקנה בסביבת העבודה
6.....	תרשים מסכים – diagram flow screens
7.....	תיאור המסכים
11.....	בסיס נתונים
11.....	הסבר על בסיס הנתונים - MongoDB
13.....	תיאור הסכמות במערכת
18.....	גישה למידע שבמאגר הנתונים
19.....	מדריך למפתח
20.....	צד לקוח - טכנולוגיות בהן השתמשתי
22.....	kesher-frontend
27.....	צד שרת – הטכנולוגיות בהן השתמשתי
28.....	kesher-backend
30.....	רפלקציה
30.....	קשיים ואתגרים שעמדו בפניי
30.....	מה קיבלתי מהעבודה על הפרויקט?
30.....	מה הייתי עושה אחרת?
30.....	סיכום ומסקנות
31.....	ביבליוגרפיה
32.....	קטעי קוד נבחרים
32.....	מסך התחברות – attendanceScreen.tsx
35.....	LoginController.js
36.....	ChildrenService.js
37.....	ReportRepository.js

מבוא

תיאור תכולת הספר

בספר זה אספר אודות פרויקט הגמר שלי, אפרט על תהליך העבודה, אציג את המדריך למשתמש ואתאר את הטכנולוגיות בהן השתמשתי בפרקים בסיס נתונים ומדריך למפתח.

הרקע לפרויקט

הפרויקט שלי נעשה בשיתוף עמותת אלווין ישראל, שנותנת שירות ליותר מ-5200 אנשים עם מוגבלות. העמותה מפעילה במסגרתה כ-30 מעונות יום לילדים בגילאי 0-3 בהם הם מקבלים טיפולים רפואיים וסביבה מותאמת.

בנוסף, לקחתי חלק בתוכנית MAX שקישרו אותי לעמותת אלווין ולצוות שלי איתו עבדתי בשיתוף פעולה על מנת להגיע לפרויקט ברמה הכי גבוהה שניתן, מאיה יוסף המעצבת ויוני דילר היזם.

תהליך המחקר

בשיחה הראשונה עם נציגי עמותת אלווין הופתעתי לשמוע שב-2021 הצוות בגנים עדיין כותב להורים במחברות קשר כמו פעם, עם דף ועט. מיותר לציין שהשיטה הזאת מלאה בחסרונות, המחברת לא זמינה בכל מקום ובכל זמן, יכולה להיהרס או ללכת לאיבוד, לא זמינה לשני ההורים במקרה של הורים פרודים/גרופים, מצריכה מהגננת לבצע עבודה נוספת במקום להשקיע את הזמן בילדים ועוד.

הופתעתי אף יותר לגלות שגם לגנים הציבוריים והפרטיים בישראל אין מערכת העונה באופן מלא על הצרכים.

[GanBook](#) - האפליקציה מציעה שיתוף תמונות בין הגנן/ת לכלל ההורים ופרסום הודעות כלליות בלבד. מעבר לממשק משתמש פחות ידידותי, האפליקציה לא עונה על הצורך של קשר אישי בין אנשי הצוות להורים.

[infogan](#) – לא קיבלתי גישה לשימוש במערכת, אבל לפי האתר וחנויות האפליקציות, גם היא, בדומה ל-GanBook לא עונה על צורך הקשר האישי וקיבלה דירוגים נמוכים מאוד מהמשתמשים. אציין כי בשוק הבין לאומי קיימים ממשקים טובים יותר אולם אין תמיכה בשפה העברית וגם הם אינם קולעים לצרכי העמותה במלואם.

אילו חידושים יש בפרויקט

- יעילות שיתוף המידע: המוצר מרכז חלק משמעותי מהתקשורת בין אנשי הצוות להורים, ומכיל בתוכו מידע רב וחיוני.
 - סדר וארגון: האפליקציה הופכת את תהליך הדיווח ממסורבל ומלא בבעיות לפשוט, מסודר וזמין בכל מקום.
 - חווית המשתמש: המוצר קל לשימוש, אינו מסורבל, וכולל הרבה אלמנטים גרפיים. לכן, השימוש במוצר יהיה פשוט גם עבור אנשים שאינם בקיאים בשימוש בטכנולוגיה.
 - התאמה אישית: המוצר שלנו נעשה בליווי צמוד של עמותת אלווין. במהלך העבודה על המוצר התקיימה תקשורת בלתי פוסקת עם העמותה וצוותים שסיפרו לנו ממקור ראשון על הבעיה. כתוצאה מכך, המוצר שלנו מותאם לצורכיהם בצורה המרבית.
- הסקיצות למסכים שהצגנו בפני אלווין:



בחירת הנושא ומוטיבציה לעבודה

בתחילת העבודה עם MAX, קיבלתי חוברת עם אוסף של כ-30 בעיות איתן עמותות שונות מתמודדות, וביניהן בעיית מחברת הקשר.

בפאן הטכנולוגי הרעיון נראה לי אפשרי ומעניין ליישום ובפאן האישי ראיתי את האימפקט שיש לו כבר מהטווח הקצר ואת הפוטנציאל הגדול בעתיד. בנוסף, בשיחות עם MAX נאמר לי שאלוין רתומים למשימה ושימחו לעזור במה שצריך, ולשמחתי באמת היה כך.

מעבר לערך המוסף עבור עמותות אלוין (האפליקציה) והידיעה שהפרויקט שלי הוא לא עוד "פרויקט למגירה", אלא אנשים באמת ישתמשו בו בעתיד, אני זכיתי לחוות תהליך של יזמות והקמת "סטארט אפ". עבדתי עם אנשים מקצועיים ומנוסים החל מהצוות שלי ועד לפגישות עם חברות ענק כמו cyberark וצברתי ניסיון שאין להרבה בני גילי.

על איזה צורך הפרויקט עונה? איזה פתרון הפרויקט הזה בא לתת?

מטרת המוצר שלי היא ליצור תמונת מצב מלאה ועדכנית אודות כל ילד בגן, ליעל את תהליך הדיווח, ולאפשר מעקב ארוך טווח אחר התקדמות הילד.

הפתרון

הפתרון לבעיה הינה מערכת מעקב ודיווח יעילה ונוחה לשימוש על מנת שהצוות יוכל להתעסק בעיקר, הילדים, וההורים יישארו מעודכנים. לכל הורה יהיה משתמש בו הוא יוכל לקבל מידע אודות הילד ולכל איש צוות משתמש עם יכולת דיווח פרטית לכל ילד.

בגרסאות הבאות

בגרסאות הבאות ארצה להוסיף תמיכה בשפות נוספות מלבד עברית (בהתאם לבקשת העמותה), להוסיף סטטיסטיקות ומעקבים, לאפשר יכולת שליפה וסינון של דיווחים, ליצור משתמש "אדמין" לניהול כל הגנים, להכין עמוד הגדרות בו כל משתמש יוכל לשנות את ההגדרות ומסך בו מתרכזות כל ההתראות עבור אנשי הצוות.

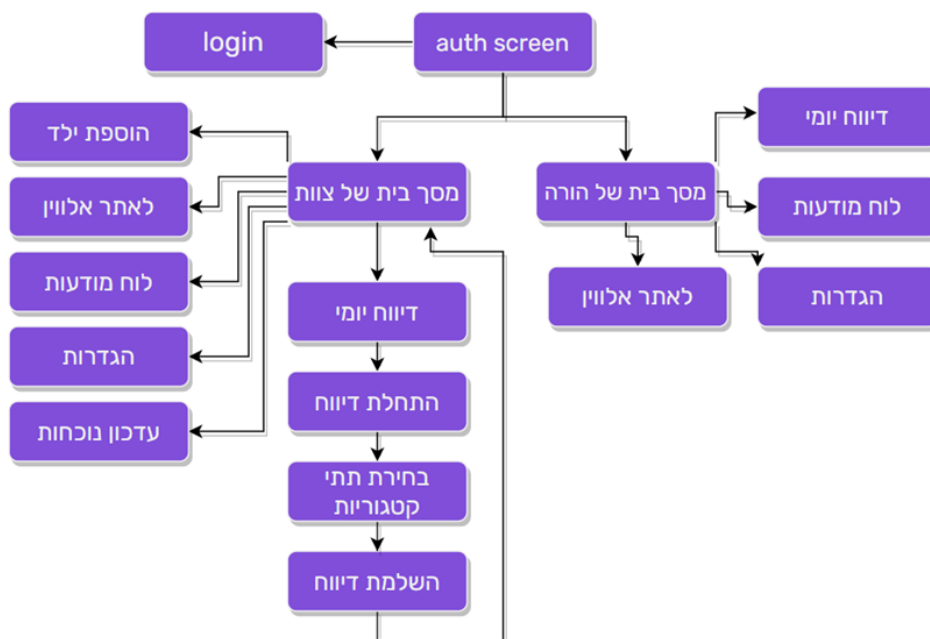
מדריך למשתמש

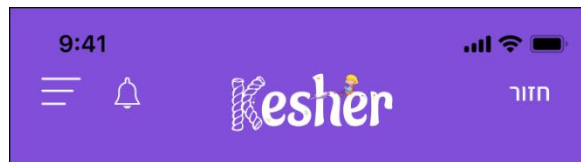
בפרק זה אציג את המסכים של האפליקציה. האפליקציה עוצבה בקפידה על ידי מאיה יוסף. על מנת להעביר חווית משתמש אולטימטיבית, אפיינו את המסכים תוך כדי שיח עם נציגים מעמותת אלווין והופעלה חשיבה על הפרטים הכי קטנים. לדוגמה, הצבע הסגול נבחר להוביל את האפליקציה משום שהוא הצבע המסמל את קהילת האנשים עם מוגבלויות.

הוראות התקנה בסביבת העבודה

ברגע, כדי להשתמש באפליקציה על המשתמש להוריד את אפליקציית expo go, להיות על אותה הרשת של השרת ולסרוק QR code מתאים על מנת לטעון את הפרויקט. אציין כי בעתיד הקרוב האפליקציה תהיה זמינה בחנויות האפליקציות.

תרשים מסכים – diagram flow screens





header

בנוסף, באמצע header יופיע שם המסך או הלוגו (בהתאם למסך) ובקצה השמאלי ביותר כפתור "חזור" במידה והמשתמש היה בעמוד אחר לפני.

אציין כי בגרסה זו לאייקון הפעמון אין משמעות אולם בהמשך, לחיצה עליו תוביל להודעות שהמשתמש קיבל.



Auth screen

המסך הראשוני במערכת. במסך נערכת בדיקה האם קיים id של שתמש ב local storages. ז"א- האם ישנו משתמש מחובר למערכת במכשיר עליו רצה האפליקציה. במידה ואכן יש משתמש – מתבצע מעבר למסך הבית של המשתמש עם המידע הדרוש (תלוי במשתמש). אחרת- מתבצע מעבר למסך login, בו המשתמש יכול להתחבר לאפליקציה.

Login screen

מסך ההתחברות למערכת. המשתמש מזין את הדוא"ל שלו והסיסמא בשדות המתאימים ולאחר מכן לוחץ על כפתור "התחברות" בתחתית המסך. במידה ותהליך ההתחברות צלח (נמצא משתמש ששם המשתמש שלו והסיסמא שלו תואמים לפרטים שהוזנו) - מתבצע מעבר למסך הבא. אחרת מופיעה התראה המעדכנת את המשתמש כי ההתחברות כשלה בעקבות הזנת סיסמא או שם משתמש לא נכונים.

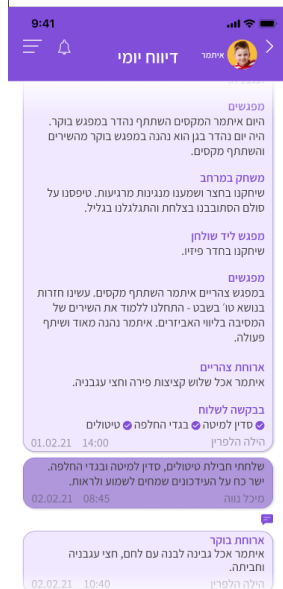


Parents home screen

מסך הבית של ההורים. במסך זה מוצג הילד אליו מתייחס המידע ורשימה נפתחת הנותנת אופציה להחליף לילד אחר במידה ויש לאותו הורה כמה ילדים במסגרות שמשתמשות באפליקציה.

בנוסף, מופיעים כפתורים המובילים למסכים הבאים לקבלת מידע נוסף.

אציין כי הצהרת הקורונה לא תיכנס בגרסה הסופית של המערכת.



Daily report screen

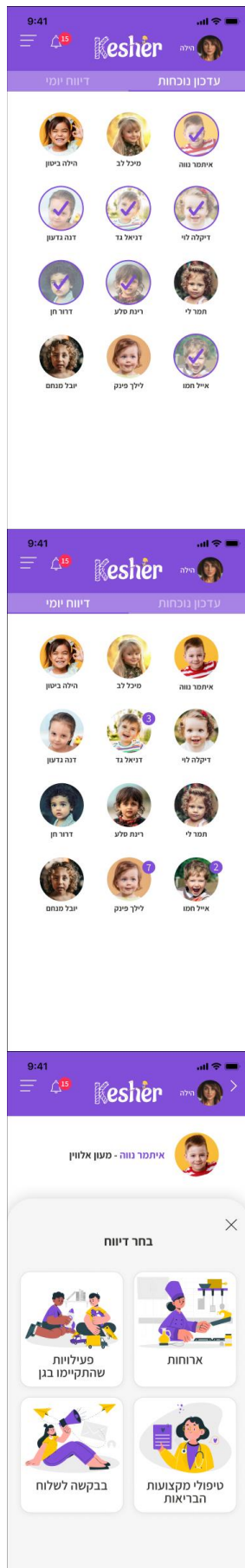
מסך דיווח יומי. מסך זה מציג להורים את כל הדיווחים היומיים אודות הילד באופן פרטי. בנוסף, ניתנת אפשרות תגובה.



Events board screen

מסך לוח מודעות. מסך זה מוצג לכל ההורים המשויכים לגן ובו מוצגות הודעות חשובות כמו פעילויות, ימי הולדת, חגים וחופשות ומועדים נוספים שהצוות יכול להוסיף בלחיצה על כפתור שיופיע בתחתית המסך.

בעת לחיצה על אירוע, יפתח פירוט אודות אותו אירוע.



Attendance screen

מסך דיווח נוכחות. במסך זה יוצגו תמונות הילדים ושמים ובלחיצה על ילד, יסמן הגנן שהילד נכח בגן באותו היום.

Daily report screen

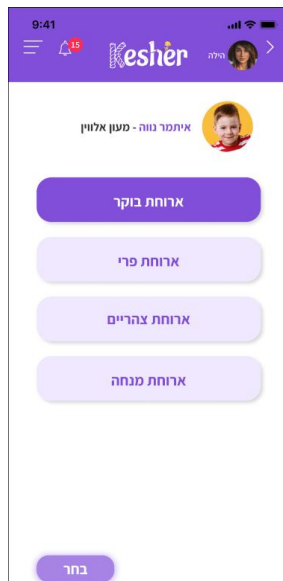
מסך תחילת דיווח נוכחות. במסך זה יוצגו תמונות הילדים ושמים. בלחיצה על ילד, יועבר המשתמש למסך חדש של תחילת דיווח.

אציין כי בגרסה זו למספרים ליד התמונות אין משמעות אולם בגרסאות הבאות הם יתריעו על תגובות של הורים לדיווחים.

Start report screen

מסך התחלת דיווח. במסך זה יוצגו שאר הדיווחים מאותו יום אם קיימים ובנוסף כפתור "התחל דיווח" בתחתית המסך. בלחיצה על הכפתור, יפתח המודל אם 4 קטגוריות אפשריות לדיווח.

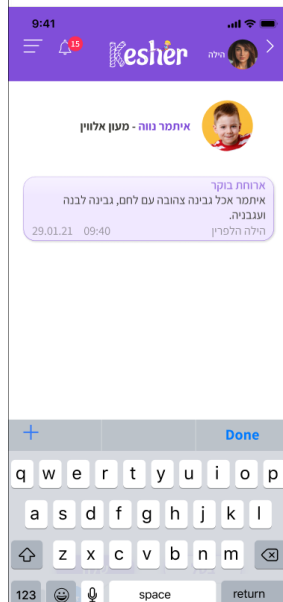
בלחיצה על אחת הקטגוריות יועבר המשתמש למסך עם תתי הקטגוריות בהתאם.



Subcategories screen

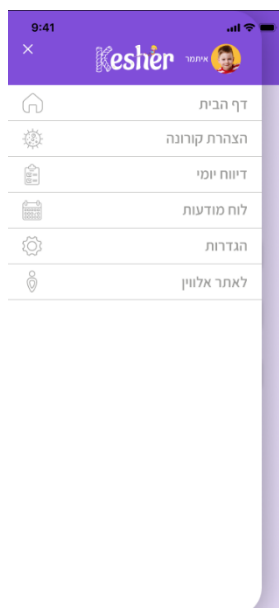
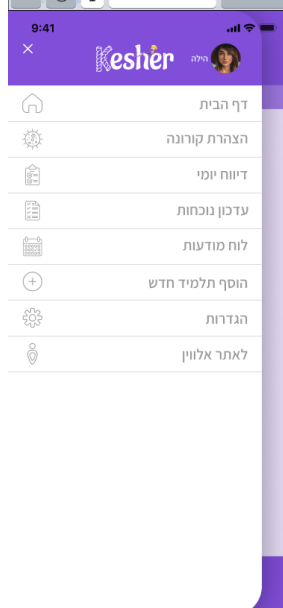
מסך בחירת תת-קטגוריות. במסך זה יוצגו תתי הקטגוריות האפשריים בהתאם לקטגוריה הראשית שהמשתמש בחר.

במסך זה ניתנת אופציה בחירה מרובה לתתי קטגוריות ובלחיצה על בחר יועבר המשתמש למסך השלמת הדיווח.



Complete report screen

מסך השלמת דיווח. במסך זה יוצגו תתי הקטגוריות שנבחרו במסך הקודם עם מקום להקלדת פירוט. בלחיצה על כפתור "סיימתי" (מוסתר על ידי המקלדת בתמונה), יושלם הדיווח והמשתמש יחזור למסך הדיווחים הראשי.



מימין, התפריט של ההורה ומשמאל של הצוות. ניתן לפתוח את התפריט בלחיצה על הכפתור השמאלי בheader או להחליק ימינה את המסך.

בסיס נתונים

הסבר על בסיס הנתונים - MongoDB

בפרויקט שלי אני משתמשת בבסיס נתונים שנקרא MongoDB (מונגו). מונגו הוא אחד ממאגרי הנתונים הנפוצים ביותר בעולם ה-NoSQL (מאגר נתונים לא טבלאי) בסיס נתונים זה נשען על מסמך JSON בניגוד לטבלה, כפי שקיים במסד נתונים ראציונלי. ב-MongoDB מבנה הנתונים JSON מכונה BJSON – Binary JSON.

בחרתי להשתמש בבסיס נתונים לא טבלאי בגלל הגמישות שהוא מאפשר בשמירת המידע והשימוש הפשוט ב-JSON, אותה שיטה שאני משתמשת לשמירת מידע גם בצד הלקוח וגם בצד השרת.

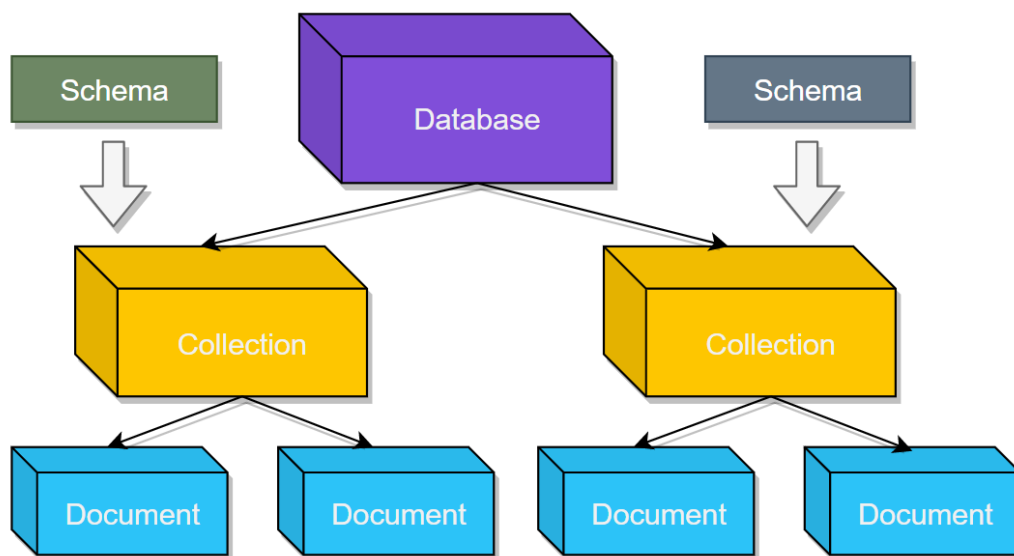
במסגרת התוכנה, ניתן לפתוח מסד נתונים שירוך על שרת של MongoDB בשם atlas.

בפרויקט שלי, הייתי צריכה לשמור סוגים שונים של מידע - מידע על אנשי הצוות הרשומים (שם משתמש, סיסמא, המעונות הקשורים אליהם וכיו"ב), דיווחים (סוג הדיווח, פירוט הדיווח, למי הוא מקושר וכיו"ב) ועוד (פירוט נוסף בתיאור הסכמות).

מאגר הנתונים מאפשר לי לשמור סוגים שונים של מידע, באמצעות Collections, אוספים שונים של מסמכים, Documents שמחלקים את המידע בצורה נוחה ויעילה. כל Document הוא למעשה מסמך JSON שנכנס לבסיס הנתונים. לדוגמה, ביצירת ילד חדש באפליקציה, יוכנס מסמך מסוג JSON ל collection של childrens ויווצר לו מספר _id ייחודי על ידי מונגו.

בפרויקט שלי קיימים חמישה Collections:

- parents - אוסף המסמכים אודות כל המשתמשים מסוג הורים ופרטיהם.
- staffs - אוסף המסמכים אודות כל המשתמשים מסוג צוות (גננים/מטפלים/אדמינים...) ופרטיהם.
- childrens - אוסף המסמכים אודות כל הילדים ופרטיהם.
- schools - אוסף המסמכים אודות כל הגנים ופרטיהם.
- reports - אוסף המסמכים אודות כל דיווח יומי.



לכל Collection שהגדרתי, הגדרתי גם Schema תואמת. Mongoose יוצר תבנית לכל מסמך שנכנס ל Collection לפי הסכמה, ובכך מאפשר לשמור על המידע בצורה מסודרת ונוחה .

תיאור הסכמות במערכת

parents

שם	תיאור	טיפוס הנתונים	האם חובה?	הערות נוספות הראויות לציון	ערך ברירת מחדל (אם יש)
_id	מספר מזהה	objectId	נוצר באופן אוטומטי	מזהה ייחודי הנוצר עבור כל מסמך באופן אוטומטי על ידי מונגו	
name	השם של המשתמש	{first: String, last: String}	כן		
address	הכתובת של המשתמש	{city: String, street: String, number: Number}	כן		
phoneNumber	מספר הטלפון של המשתמש	Number	כן		
email	המייל של המשתמש	String	כן	משמש גם בהזדהות בכניסה לאפליקציה	
password	הסיסמה של המשתמש	String	כן		
schools	הגנים מיושבים למשתמש	Array[objectId]	לא		
children	רשימה עם הילדים של המשתמש	Array[objectId]	לא		
active	האם המשתמש פעיל או לא, במקום מחיקה	Boolean	כן		true

שם	תיאור	טיפוס הנתונים	האם חובה?	הערות נוספות הראויות לציון	ערך ברירת מחדל (אם יש)
_id	מספר מזהה	objectId	נוצר באופן אוטומטי	מזהה ייחודי הנוצר עבור כל מסמך באופן אוטומטי על ידי מונגו	
name	שם המעון	String	כן		
address	הכתובת של הגן	{city: String, street: String, number: Number}	כן		
eventsBoard		Array[{ title: String, details: String, startTime: Date, endTime: Date, createdDate: Date, creatorId: objectId }]	לא		
parents	רשימה של ההורים המקושרים למעון	Array[objectId]	לא		
children	רשימה של ילדים המקושרים למעון	Array[objectId]	לא		
staff	רשימה של עובדים המקושרים למעון	Array[objectId]	לא		
active	האם המעון פעיל או לא, במקום מחיקה	Boolean	כן		true

שם	תיאור	טיפוס הנתונים	האם חובה?	הערות נוספות הראויות לציון	ערך ברירת מחדל (אם יש)
_id	מספר מזהה	objectId	נוצר באופן אוטומטי	מזהה ייחודי הנוצר עבור כל מסמך באופן אוטומטי על ידי מונגו	
name	שם הילד	{first: String, last: String}	כן		
profilePic	תמונה של הילד	String	לא, אם כי יש ברירת מחדל		תמונה ברירת מחדל
birthDate	תאריך הילדה של הילד	Date	כן		
school	הגן של הילד	objectId	לא		
active	האם המשתמש פעיל או לא, במקום מחיקה	Boolean	כן		true

שם	תיאור	טיפוס הנתונים	האם חובה?	הערות נוספות הראויות לציון	ערך ברירת מחדל (אם יש)
_id	מספר מזהה	objectId	נוצר באופן אוטומטי	מזהה ייחודי הנוצר עבור כל מסמך באופן אוטומטי על ידי מונגו	
name	שם חבר הצוות	{first: String, last: String}	כן		
address	הכתובת של המשתמש	{city: String, street: String, number: Number}	כן	הכתובת של המשתמש	
role	תפקיד חבר הצוות (גנן, מטפל, אדמין...)	String	כן		
profilePic	תמונה של חבר הצוות	String	לא		
birthDate	תאריך לידה	Date	כן		
phoneNumber	מספר טלפון		כן		
email	דוא"ל	String	כן	משמש גם כהזדהות בבניסה לאפליקציה.	
password	סיסמה של חבר הצוות	String	כן		
schools	מעונות המקושרים לחבר הצוות	Array [objectId]	לא		
active	האם המשתמש פעיל או לא, במקום מחיקה	Boolean	כן		true

שם	תיאור	טיפוס הנתונים	האם חובה?	הערות נוספות הראויות לציון	ערך ברירת מחדל (אם יש)
_id	מספר מזהה	objectId	נוצר באופן אוטומטי	מזהה ייחודי הנוצר עבור כל מסמך באופן אוטומטי על ידי מונגו	
date	היום אליו מקושר הדיווח	Date	כן		
Child	הילד אליו מקושר הדיווח	objectId	כן		
attendance	האם הילד נכך באותו יום בגן	Boolean	כן		false
subReport	דיווח יומי אישי על ידי צוות המעון	Array[{date: Date, creator: objectId, reportId: objectId, message: string}]	לא		
comments	תגובות ההורים לדיווח	Array[{date: Date, creator: objectId, category: String, subcategory: String, details: string}]	לא		

גישה למידע שבמאגר הנתונים

ישנם מספר מודולים שמטפלים בחיבור בין בסיס הנתונים לשרת, אני בחרתי לעבוד עם mongoose.

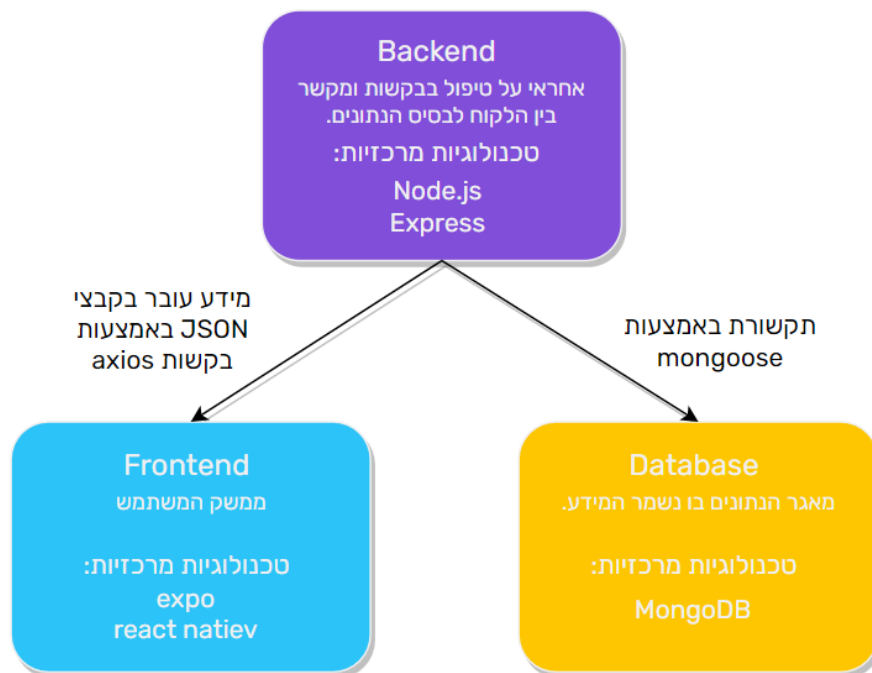
mongoose היא ספריית ODM (Object-Document Mapping), המותאמת לעבודה בסביבה א-סינכרונית, ומסייעת בהעשרת היכולות בשימוש במונגו או במידול הנתונים, תוך סיפוק פתרון פשוט המבוסס על הסכמות כדי למדל את המידע.

לדעתי, השימוש ב mongoose הוא יעיל וכדאי בגלל סיבה אחת עיקרית: הוא מאפשר גישה קלה למאגר הנתונים תוך שימוש בפעולות נוספות, הייחודיות רק לו למען עריכת המידע במונגו.

דוגמאות לשיטות מידול שמציע mongoose:

- find() - שיטה המחזירה את כל המסמכים מאוסף מסוים. שיטה זו מאפשרת להעביר פרמטר חיפוש, ובכך להחזיר רק את המסמכים הרלוונטיים לפרמטרים.
- findById() - שיטה המחזירה מסמך מסכמה מסוימת לפי ה-id של המסמך.
- remove() - שיטה המשמשת למחיקת מסמך מסוים לפי תנאי נתון. במידה ולא מסופק תנאי, השיטה תמחק את כל המסמכים.
- save() - שיטה המשמשת להכנסת מסמך מסוים לאוסף.

מדריך למפתח



הפרויקט שלי מורכב משתי תיקיות:

kesher-backend המכילה את הקבצים הרלוונטיים לשרת ולמסד הנתונים.

kesher-frontend המכילה את הקבצים הרלוונטיים לצד הלקוח.

בחלק זה, אפרט אודות הקבצים תוך התייחסות לתיקיות המרכזיות - התייחסות נפרדת לצד הלקוח ולשרת.

צד לקוח - טכנולוגיות בהן השתמשתי

לאפליקציית "קשר" יש חשיבות רבה לחוויית המשתמש משום שהיא באה להחליף את מחברת הקשר המוכרת, וצריכה להיות נוחה ופשוטה לאוכלוסייה מגוונת.

אציין כי כתבתי את המסכים בTypescript, אולם בגלל חוסר הזמן וחוסר הידע המקדים לא הצלחתי למצות את הפוטנציאל והיכולות של השפה ואם היה לי זמן הייתי משכתבת את הקבצים ונותנת למשתנים טיפים יותר מפורטים.

הטכנולוגיות בהן השתמשתי:

React native - I React

React היא ספרייה בתוך JS, אשר משמשת לכתיבת ממשקי משתמש. React מאפשרת שילוב בין Node.js לבין שפות XML, המבוססת על שימוש במחלקות ובפונקציות ובכך מאפשרת שמירת מידע וקלט מהמשתמש וניתוחו בפעולות ב-JS כמו לדוגמה, קריאה לשרת.

React מבוססת על רכיבים (Components). כאשר כל רכיב מציג חלק במסך וניתן להשתש ברכיבים הקיימים או ליצור חדשים. לכל רכיב יש תכונות ופעולות משלו, ולרוב הרכיבים יש מספר פעולות ותכונות זהות המוגדרות בספרייה. קיימים שני סוגי מידע השולטים ברכיב:

state - דרך לשמירת מידע דינאמי ושמירת מצבים. השימוש ב-state הוא נוח ויעיל משום שאין צורך ליצור תכונות אחרות לשמירת נתונים או פעולות להוצאת הנתונים מהמאגר. מה ההבדל בין let state? בReact המסך מתרנדר רק כשצריך. אם נרצה לבצע שינוי שגם נוכל לראות, נשתמש בstated הוא יעדכן את הDOM שבוצע שינוי וצריך לבצע רנדור. בlet המידע ישתנה אך לא נראה את על המסך.

props - קיצור של המילה properties באנגלית. הם שימושיים ואף הכרחיים בעת יצירה ושימוש ברכיב, והם מאפשרים מעבר מידע ותמרון שלו. אחד מהשימושים הנפוצים של props הוא הגדרת "מאפיינים" מסוימים לרכיב, שיאפשרו שליטה בתוכן שלו בעת השימוש ברכיב באמצעות העברת props כפרמטר (לדוגמה: שם של כפתור).

React Native היא ספריית קוד פתוח, שבעזרתה ניתן ליצור אפליקציה אחת שתואמת למספר מערכות הפעלה שונות, במקום לפתח במקביל מספר גרסאות של אותה האפליקציה.

React Native מאפשרת למפתחים להשתמש ב-React יחד עם יכולות של פלטפורמת native. עקרונות העבודה של React Native זהים כמעט לגמרי לאל של React, כאשר ההבדל המרכזי הוא

אופי הרכיבים- רכיבי native לעומת רכיבי web, הבדל שנגזרים ממנו, מן הסתם, הבדלים נוספים המשפיעים על כתיבת הקוד .

Expo

Expo היא פלטפורמה עבור פיתוח אפליקציות React אוניברסליות. היא נותנת כלים ושירותים המבוססים על React native ופלטפורמות native אחרות כדי לעזור בפיתוח הקוד, וכדי לספק הרצה מהירה שלו על iOS, Android, ואפליקציות web .

Expo נוחה ושימושית במיוחד עבור מפתחים מתחילים, מאחר והיא מאפשרת ליצור פרויקט עם managed-workflow. בעת השימוש ב- managed-workflow, המפתח לא צריך להשתמש ב-Xcode או ב-Android studio כדי לפתח את האפליקציה, הוא יכול פשוט לכתוב קוד JavaScript ולשלט בקונפיגורציה של דברים כמו אייקון האפליקציה דרך הקובץ (app.json).

expo גם מאפשרת למפתח להשתמש ביכולות של המכשיר עצמו כמו מצלמה, אימות ביומטרי, מערכת הקבצים ועוד.

השתמשתי בכלים השונים ש-expo נותנת כדי לפתח את האפליקציה ב-javascript. בנוסף, השתמשתי באפליקציה expo go כדי להריץ את הקוד ישירות על הטלפון שלי.

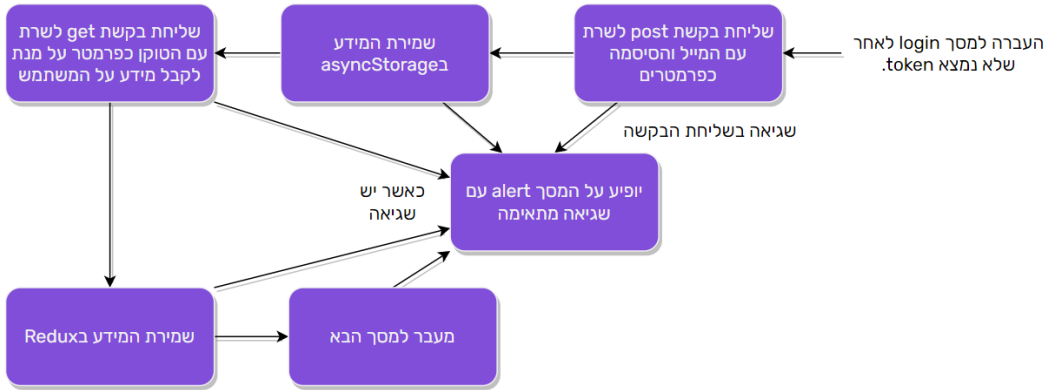
חשוב לדעת

Hooks הם תוספת חדשה חדשה יחסית בReact והם מאפשרים להשתמש בstate ובתכונות של React מבלי לכתוב מחלקה. הhooks העיקריים בהם נשתמש:

- **useEffect** – על ידי השימוש בה אנחנו אומרים לReact שהקומפוננטה צריכה לבצע משהו לאחר הרנדור (מאין "תופעת לוואי" של הרנדור, sideeffect) מסוים המועבר בסוגריים המרובעים. נשתמש בה לדוגמה כאשר נרצה לקבל מידע מהשרת פעם אחת, כשמהסך עולה, ולא אחרי כל רנדור. במקרה כזה נשאיר את הסוגריים המרובעים ריקים מפני שאנחנו לא רוצים שתתבצע עוד פעם. תבנית של `useEffect`:
`useEffect(() => {}, [])`
- **useState** – הReact לא מרנדור את המסך באופן תמידי, אלא רק כאשר מתבקש בכך. מתי הוא מתבקש? כאשר הstate משתנה. לדוגמה, אם נשמור מידע בconst רגיל ונשנה את הערך שלו המסך לא יתעדכן. לכן, נצטרך לשמור את המידע בתוך state ולעדכן אותו (את המידע) באמצעות `setState()`.
יצירת משתנה `useState` חדש: `const [state, setState] = useState()`

kesher-frontend

expo	לאחר יצירת פרויקט expo עם managed-workflow, נוצרת תיקייה בשם expo. התיקייה מכילה מידע ש-expo צריך על מנת לספק את הכלים שלו, ומכילה metadata וקבצי קונפיגורציה רלוונטיים.
expo-shared	לאחר יצירת פרויקט expo עם managed-workflow, נוצרת תיקייה בשם expo-shared. המכילה קובץ שנקרא assets.json. תיקייה זו מכילה את חלק מהתמונות הרלוונטיות לאפליקציה "דחוסות" בגודל האופטימלי ביותר.
assets	תיקיה המכילה בתוכה קבצים הקשורים לעצוב האפליקציה.
	fonts - תיקיה המכילה את הקבצים של הגופנים.
	icons - תיקיה המכילה קבצים עם האייקונים שמופיעים באפליקציה (שמורים כרכיבים בעזרת SVG).
	images - תיקיה המכילה את כל התמונות באפליקציה כמו האייקון או הלוגו שמופיע בheader.
	globalStyles.tsx - קובץ המכיל מידע לגבי עיצוב האפליקציה כך שבמקרה של שינוי הצבע הראשי או הגופן, נצטרך לשנות שורה אחת בקובץ אחד במקום לעבור על כל הקבצים.
components	בתיקיה זו נמצאות כל הקומפונינטות, כל הרכיבים שמרכיבים את המסכים השונים. לרובם אין פונקציונליות מיוחדת והם בעיקר מציגים מידע, ולכן אסביר קובץ אחד.
	subCategoryButton.tsx - הקומפונינטה מקבלת כprops text בהתאם לטקסט שרוצים שיוצג על גבי הכפתור. Picked מקבל ערך בוליאני בהתאם למצב הכפתור, על מנת לקבוע את העיצוב שיוצג, כהה יותר לנבחר ובהיר ללא נבחר. בנוסף, נקבל פרמטר onPress שמקבל את הפונקציה שנרצה שתרוץ כאשר הכפתור נלחץ.
navigation	תיקיה המכילה את כל הקבצים האחראים לניווט בין העמודים באמצעות ספריית React Navigation. לכל הקבצים מבנה כמעט זהה ואין בהם לוגיקה, לכן, אסביר על קובץ אחד.
	mainDrawer.tsx משתנה: role מקבל את סוג המשתמש מה Redux על מנת להציג לו את המסכים המתאימים. מגדירים "תפריט מגירה" באמצעות השורה <code>const Drawer = createDrawerNavigator();</code> התאג <code><Drawer.Navigator></code> מקבל לתוכו props הקשורים בהגדרות ועיצוב התפריט ובתוכו <code><Drawer.Screen></code> אשר מגדירים את המסכים אליהם ניתן לעבור מהתפריט והגדרות נוספות כמו ההדר שיוצג בהם או השם שלהם.
node-modules	לתיקיה זו מגיעות כל החבילות שמורידים באמצעות npm או yarn.

<p>תיקיה זו מרכזת את כל המסכים באפליקציה. ארחיב כאן על המסכים עם הפונקציונליות המסובכת ביותר.</p>	<p>screens</p>
<p>loginScreen.tsx</p> <p>מסך ההתחברות לאפליקציה.</p> <p>משתנים:</p> <ul style="list-style-type: none"> email הוא משתנה מסוג useState ומיועד לemail אותו יכניס המשתמש על מנת להתחבר. password הוא משתנה מסוג useState ומיועד לסיסמה אותה יכניס המשתמש על מנת להתחבר. אציין כי בתוך השדה בו מכניסים סיסמה מוגדר כsecureTextEntry . <p>פונקציות:</p> <ul style="list-style-type: none"> storeData() – הפונקציה מקבלת token ושומרת אותו בasyncStorage. handleParentLoginRequest () – הפונקציה נקראת כאשר לוחצים כל כפתור "התחבר כהורה". בשלב הראשון היא שולחת לשרת בקשה עם המייל והסיסמה על מנת לבדוק כי המשתמש באמת קיים. אם כן, מחזירה טוקן אותו היא שומרת בasyncStorage ושולחת עוד בקשה לשרת על מנת לקבל פרטים על המשתמש (שם, תפקיד, ילדים משויכים) אותם היא שומרת בRedux. ולבסוף עוברת לאפליקציה עצמה. אם המשתמש לא קיים/קיימת בעיית התחברות, מופיע alert על המסך והשדה של הסיסמה נמחק. handleStaffLoginRequest() – כמו הפונקציה הקודמת רק מותאמת למידע שצריך עבור משתמש מסוג "צוות". 	
	<p>attendanceScreen.tsx</p> <p>מסך דיווח נוכחות.</p> <p>משתנים:</p>

<ul style="list-style-type: none"> • DATA הוא משתנה מסוג <code>useState</code> ומכיל בתוכו רשימה של אובייקטים כאשר כל אובייקט הוא מידע על ילד: שם, תמונה, נוכחות <code>id</code>. <p>פונקציות:</p> <ul style="list-style-type: none"> • <code>useEffect()</code> – הפונקציה מחולקת לשלושה חלקים ותפקידה הוא להחזיר את DATA מוכן לשימוש. בחלק הראשון היא שולחת בקשה לשרת ומקבלת רשימה של ילדים המשויכים לאותו הגן. בשלב השני, היא לוקחת מכל אובייקט של ילד את ID שלו, ושולחת בקשה חדשה לשרת עם מידע על הנוכחות של הילד באותו היום. בשלב השלישי והאחרון, אני מסדרת את כל המידע באמצעות <code>forEach</code> (מצמידה לכל אובייקט של ילד את המידע אודות הנוכחות) כדי לקבל אובייקט עם כל המידע הדרוש ושמה אותו בDATA. • <code>handleDATAChange(item)</code> – הפונקציה אחראית על טיפול בנכחות/הסרת נוכחות של ילד. היא מקבלת לתוכה <code>item</code>, שהוא בעצם האובייקט הנבחר מDATA שהגדרנו בפונקציה הקודמת. היא מעדכנת את DATA עצמה בשינוי (<code>true</code> או <code>false</code> להפך) כדי שהעיצוב ישתנה בהתאם, ושולחת בקשה לשרת לשנות בבסיס הנתונים את המידע. 	
<p>reportSubCategoryScreen.tsx</p> <p>מסך בחירת תתי הקטגוריות לדיווח.</p> <p>משתנים:</p> <ul style="list-style-type: none"> • DATA הוא משתנה מסוג <code>useState</code> ומכיל בתוכו רשימה של אובייקטים כאשר כל אובייקט הוא תת קטגוריות באותה קטגוריה שהמשתמש בחר במסך הקודם אם מידע נוסף כמו האם נבחר או לא וה <code>id</code> שלו. • <code>ACTIVITIES_DATA</code> – מכיל בתוכו את כל תתי הקטגוריות האפשריים שהוגדרו מראש (קיים משתנה כזה לכל קטגוריה). <p>פונקציות:</p> <ul style="list-style-type: none"> • <code>getData()</code> – מקבלת את הקטגוריה שנבחרה ומחזירה את המשתנה ששומר את המידע אודות תתי הקטגוריות. • <code>useEffect()</code> – הפונקציה מסדרת את כל המידע אודות הקטגוריה שנבחרה ברשימה של אובייקטים ומוסיפה לו את השדה <code>selected = false</code> כדי לקבל אובייקט עם כל המידע הדרוש ושמה אותו בDATA. • <code>handleItemPress(item)</code> – הפונקציה אחראית על סימון פריט כנבחר/הסרתו מנבחר. היא מקבלת לתוכה <code>item</code>, שהוא בעצם האובייקט הנבחר מDATA שהגדרנו בפונקציה הקודמת, ומעדכנת את DATA עצמה בשינוי (<code>true</code> או <code>false</code> להפך) כדי שהמידע ישמר והעיצוב ישתנה בהתאם. 	

<ul style="list-style-type: none"> • <code>selectedCategories()</code> – יוצרת רשימה חדשה רק עם הפריטים שנבחרו על מנת להעביר למסך הבא. • <code>handleSubmitAndNext()</code> – נקראת כאשר לוחצים על כפתור הסיום. מאכסנת את המידע ב Redux ועוברת למסך הבא. 	
<p>complteReportScreen.tsx</p> <p>מסך השלמת הדיווח.</p> <p>משתנים:</p> <ul style="list-style-type: none"> • <code>subCategories</code> הוא משתנה מסוג <code>useState</code> ומכיל בתוכו רשימה של תתי הקטגוריות ששמרנו בעמוד הקודם ב Redux ואת התוכן שלהם בתור אובייקטים. <p>הואפונקציות:</p> <ul style="list-style-type: none"> • <code>useEffect()</code> – הפונקציה מסדרת את כל המידע אודות תתי הקטגוריה ומוסיפה לכל אובייקט את השדה <code>report_value</code> שיכיל את הפירוט שאיש הצוות כותב. ולבסוף, הרשימה החדשה מוכלת ב <code>subCategories</code>. • <code>handleInput(item, input)</code> – הפונקציה מקבלת את <code>item</code> שמקלידים בו ואת ערך ההקלדה ומעדכנת אותו על מנת שישמר ונוכל לראות אותו על המסך. • <code>handleSubmit()</code> – הפונקציה מקבלת את הערך הסופי של כל הדיווח, שולחת אותו לשרת וחוזרת לעמוד הראשי. 	
<p>קובץ המסמן ל-<code>git</code> אילו קבצים לא לעדכן כאשר עושים <code>commit</code>.</p>	<p>.gitignor</p>
<p>הקובץ שמרכז בתוכו את כל פעולות הקריאה לשרת.</p> <p>הפונקציה <code>optios()</code> מקבלת את הטוקן של המשתמש מה <code>asyncStorage</code> ומוסיפה אותו להדר כדי שאחר כך, כשהבקשה מגיעה לשרת, הוא ידע שהוא נותן מידע למשתמש מחובר ולא למישהו שסתם שולח בקשות. מהקובץ מיוצאות 5 פונקציות כאשר כל אחת מרכזת בתוכה את הפונקציות באותה קטגוריה (התחברות, הורים, ילדים, צוות וגן).</p>	<p>api.tsx</p>
<p>קובץ קונפיגורציה מרכזי באפליקציה, בו נעשית הקונפיגורציה עבור חלקים באפליקציה שלא שייכים לקוד. למשל, הגדרת מסך ה-<code>"splash"</code> או האייקון של האפליקציה.</p>	<p>app.json</p>
<p>הקובץ ה"ראשי", "נקודת הפתיחה" לאפליקציה עצמה. בקובץ זה אני פותחת <code>store</code> בוא נשמר המידע מה Redux ומובילה את המשתמש לאפליקציה עצמה אם הוא מחובר, ואם לא, למסך <code>Login</code>.</p> <p>פונקציית <code>getData()</code> – הפונקציה מקבלת מה <code>asyncStorage</code> את הטוקן של המשתמש, ובכך בעצם בודקת אם הוא מחובר, אם לא קיים אחד – המשתמש עובד למסך ההתחברות. אם קיים,</p>	<p>App.tsx</p>

מבקשים מידע מהשרת אודות אותו משתמש ומכילים אותו במשתנה <code>getMeResponses</code> . כדי שנוכל לגשת אליו מכל המסכים, נעשה <code>dispatch</code> למידע.	
קובץ קונפיגורציה, "מתרגם" את כל קבצי ה-JS מסטנדרט של ES6+ ל-ES5, בו קיימת תמיכה בכל הדפדפנים.	babel.config.js
קובץ הנוצר בפעם הראשונה לאחר ההתקנה של <code>dependencies</code> לפרויקט. בזמן ההתקנה, כל ה- <code>dependency tree</code> מחושב ונשמר לקובץ ה- <code>lock</code> , יחד עם <code>metadata</code> אודות ה- <code>dependency</code> כמו למשל הגרסה של החבילה שאמורה להיות מותקנת.	package-lock.json
קובץ זה הוא סוג של "מניפסט" עבור הפרויקט. הוא מכיל <code>metadata</code> מגוון, נתונים אודות נתונים. קובץ זה משמש כמקום אחסון מרכזי עבור כלי קונפיגורציה, ותיעוד ה- <code>dependencies</code> של הפרויקט.	package.json
הקובץ הזה אחראי על השימוש ב- <code>Redux</code> . <code>Redux</code> היא ספרייה לניהול מצבים. לדוגמה, אני משתמשת בה כדי לשמור מידע על משתמש על מנת שאוכל לקבל גישה מכל מסך לפרטים בסיסיים כמו השם שלו, תמונת הפרופיל, הסוג שלו, ועוד.	reducer.tsx
קובץ קונפיגורציה עבור <code>TypeScript</code> .	tsconfig.json
שומר את הלוגים של כל השגיאות של מנהל החבילות <code>yarn</code> .	yarn-error.log
<code>Yarn</code> הוא מנהל חבילות שמשמש גם כמנהל פרויקט. הוא מאפשר להשתמש ולשתף קוד עם מפתחים אחרים מרחבי העולם, באופן בטוח, מהיר ואמין. הקוד המשותף מועבר באמצעות חבילה, אשר מכילה את כל הקוד ששותף, נוסף לקובץ <code>package.json</code> שמתאר את החבילה. על מנת לאפשר התקנות עקביות לאורך כל המערכת, <code>yarn</code> צריך יותר מידע מעבר ל- <code>dependencies</code> אשר הוגדרו בקובץ ה- <code>package.json</code> . <code>Yarn</code> צריכה לאחסן בדיוק איזו גרסה של כל <code>dependency</code> שבמערכת הותקנה. כדי לעשות זאת, <code>Yarn</code> משתמשת בקובץ <code>yarn.lock</code> .	yarn.lock

צד שרת – הטכנולוגיות בהן השתמשתי

Node.js

Node.js היא סביבת ריצה לשפת JavaScript שנכתבה ב-C++ ומבוססת על מנוע V8. Node.js מאפשרת לקוד JavaScript לרוץ ללא צורך בדפדפן, מה שמאפשר לה לשמש כשפת צד שרת.

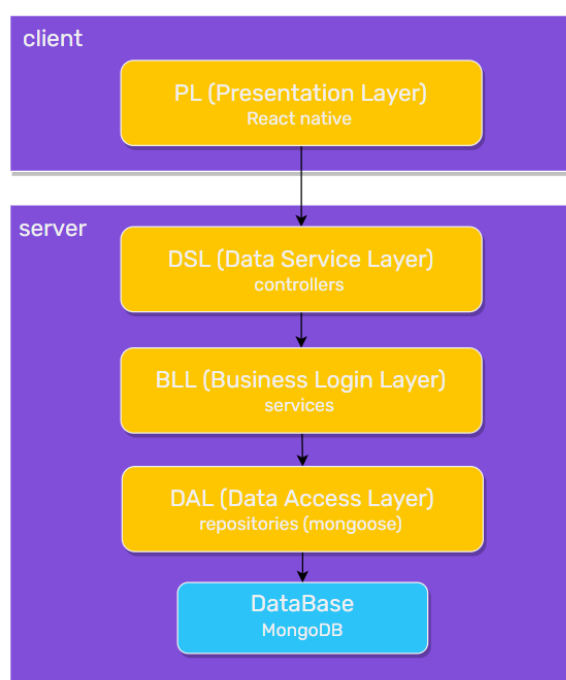
בחרתי בNode.js משום שאת צד הלקוח כתבתי בTS (מבוססת על JS) ובכך אני מקבלת התממשקות פשוטה ויעילה, העברת מידע באמצעות אובייקטים של JS והצורך להתמקצע רק בשפת תכנות אחת. בנוסף, יש לה הרבה מודולים ותוספים שיכולים לעזור בפיתוח ולחסוך זמן.

Express

Express היא ספרייה בתוך Node.js שנותנת לנו כלים פשוטים ויעילים לבתיבת שרת. לדוגמה, באמצעות Router ניתן לטפל בבקשות http שונות בצורה יעילה באמצעות ניתוב לcontroller המתאים שיטפל בהן.

ארכיטקטורת ארבע השכבות

בחרתי בארכיטקטורת ארבע השכבות ליישום המערכת. בשיטה זו אני יוצרת הפרדה בין כל שכבה כך שניתן לבצע שינויים לשכבה מסוימת, לדוגמה, לטכנולוגיה אחרת, בלי שתפגע המערכת כולה וללא צורך לשכתב את כל התוכנה מחדש. המודל נותן לנו יישום גמיש וניתן לשימוש חוזר של הפונקציות.



kesher-backend

תיקיה האחראית על פונקציות ההתחברות למערכת.	auth
<p>auth.js</p> <p>פונקציות:</p> <ul style="list-style-type: none"> generateAccessToken(user) – הפונקציה מקבלת משתמש ויוצרת לו טוקן. authenticateToken(req, res, next) – הפונקציה מקבלת בתור פרמטרים מידע על המשתמש ומאמתת אם הוא אכן קיים. 	
מכיל בתוכו רק את הקובץ <code>www</code>	bin
קובץ קונפיגורציה בה מוגדר מידע חשוב אודות השרת כמו בסיס הנתונים והקו עליו הוא רץ.	
בתיקיה זו נמצאים כל הקבצים שמנהלים את עמדות הקצה. כל קובץ הוא עמדת קצה ראשית שמורכב ממספר עמדות קצה, כאשר כל אחד מטפל בבקשה המתאימה באמצעות קריאה לservice המתאים. כל קובץ אחראי על מודול נפרד.	controller
תיקיה זו מכילה קבצים כאשר בכל קובץ מוגדרת סכמה ומיוצא מודל. כל הקבצים בנויים באותה צורה ולכן אדגים על קובץ אחד.	models
<p>ChildModel.js</p> <p>מייבאים את mongoose בשביל תבנית של סכמה. יוצרים משתנה מסוג סכמה וקובעים בתוכו את השדות ומידע לגביהם (type, האם נדרשים וערך ברירת מחדל).</p> <p>לבסוף מגדירים את הסכמה כמודל של mongoose ומייצאים אותו.</p>	
לתיקיה זו מגיעות כל החבילות שמורידים באמצעות npm או yarn בדומה לצד הלקוח.	node_modules
תיקיה זו מכילה את הקבצים האחראיים על שליפה ואחסון המידע מבסיס הנתונים.	repositories
תיקיית services מכילה בתוכה את הקבצים האחראיים על העברת מידע מהcontroller לrepository ותפקידה לשלוט בפונקציונליות של היישום על ידי ביצוע עיבוד הנתונים הפרטני. כל קובץ אחראי על מודול נפרד.	services
בקובץ זה מגדירים את כל משתנה הסביבה בשביל זמן הפיתוח.	.env
הקובץ ה"ראשי" של השרת. מגדיר את כל השירותים והקונטרולים בהם משתמש השרת.	app.js
קובץ הנוצר בפעם הראשונה לאחר ההתקנה של dependencies לפרויקט. בזמן ההתקנה, כל ה dependency tree מחושב ונשמר לקובץ ה-lock, יחד עם metadata אודות ה-dependency כמו למשל הגרסה של החבילה שאמורה להיות מותקנת.	package-lock.json

<p>קובץ זה הוא סוג של "מניפסט" עבור הפרויקט. הוא מכיל metadata מגוון, נתונים אודות נתונים. קובץ זה משמש כמקום אחסון מרכזי עבור כלי קונפיגורציה, ותיעוד dependencies של הפרויקט.</p>	<p>package.json</p>
---	----------------------------

רפלקציה

קשיים ואתגרים שעמדו בפניי

את רוב הפלטפורמות בהן השתמשתי לפיתוח הפרויקט לא הכרתי/יצא לי להשתמש לפני. הדבר הצריך ממני למידה מ-0 כיצד להשתמש בהן בצורה הנכונה והיעילה ביותר, והיום, אחרי כמה חודשי פיתוח אני חושבת שהצלחתי להגיע לרמה גבוהה ותוצר שאני גאה בו.

קושי נוסף שאני שמחה שיצא לי לחוות הוא דווקא בהצלחת הפרויקט. ככל שהתהליך התקדם עם MAX הם בחרו בנו פעמיים כאחד מתוך ששת הצוותים הכי מתקדמים בתוכנית. כל בחירה כזאת שמה אותי מחוץ לאזור הנוחות שלי וגרמה לי להתמודד עם סיטואציות שונות וביניהן עבודת צוות וניהול.

מה קיבלתי מהעבודה על הפרויקט?

למרות שצברתי המון ידע טכני והבנה נכונה של תכנון מערכות בעבודה על הפרויקט כמו מה בא לפני מה ואיך להתאים טכנולוגיות למטרה הסופית, קיבלתי הרבה יותר לחיים האמיתיים.

ראשית, היה לי לקוח אמיתי עם רצונות וצרכים ולא הייתי הלקוח של עצמי. עבדתי עם מעצבת מנוסה שכבר 20 שנה במקצוע ויודעת לאפיין את העיצוב כדי שיתאים למשתמש, חקרתי על עניינים משפטיים שרלוונטיים לאפליקציה (לדוגמה, גיליתי שיש חוק האוסר על איסוף מידע על אנשים, תעודות זהות ומידע רפואי בפרט, ללא הצדקה ורישום במאגר של המדינה).

בנוסף, יצא לי לנהל ולעמוד בסיטואציות מורכבות (שלא אשתף כאן) בהן הייתי צריכה למצוא דרך לקבל את מה שאני רוצה בסופו של דבר ולא לוותר.

מה הייתי עושה אחרת?

אני חושבת שבהלך העבודה הייתי צריכה לסמוך יותר על עצמי. עד פסח הייתי בקצב טוב ובטוחה בעצמי, ורצף של אירועים קצת קטע את העבודה. אם הייתי חוזרת אחורה לאותה תקופה, הייתי יותר אסרטיבית וסומכת יותר על עצמי.

סיכום ומסקנות

אני מאוד מרוצה מהפרויקט הסופי. תהליך העבודה על הפרויקט היה מורכב, וכלל עליות וירידות, תחושות שמחה וסיפוק לצד רגעים קשים. אני מרגישה שעברתי תהליך מסוים לאורך פיתוח הפרויקט, התייעצתי עם המון אנשים ושמעתי דעות וגישות שונות. אני שמחה שניתנה לי ההזדמנות לקחת חלק בפרויקט המיוחד הזה ואני מקווה שהוא ישמש אנשים כמה שיותר מהר.

ביבליוגרפיה

[הסבר ותיעוד של React](#)

[הסבר ותיעוד של React native](#)

[הסבר ותיעוד של Expo](#)

[הסבר ותיעוד של React Navigation](#)

[הסבר ותיעוד של Redux](#)

[הסבר ותיעוד של Async Storage](#)

[הסבר ותיעוד של mongoose](#)

[הסבר על מונגו ובסיסי נתונים לא-רציונליים מתוך אתר GeetTime](#)

[הסבר ותיעוד של axios](#)

[הסבר על Node.js](#)

[הסבר ותיעוד של jsonwebtoken](#)

[הסבר על מודל ארבע השכבות](#)

קטעי קוד נבחרים

מסך התחברות – attendanceScreen.tsx

```
import React, { useEffect, useState } from "react";
import {
  StyleSheet,
  Text,
  View,
  FlatList,
  TouchableOpacity,
  Image,
} from "react-native";
import globalStyles from "../../assets/globalStyles";
import Icons from "../../assets/icons/icons";
import { connect } from "react-redux";
import api from "../../api";

function AttendanceScreen(props: any) {
  const [DATA, setDATA] = React.useState([]);

  // ANCHOR get data from server about the children in the
  school
  // and their attendance and merge them together.
  useEffect(() => {
    const getData = async () => {
      const childrenResponse = await api
        .schools()
        .getChildren(props.user.schools[0]);

      let ids: Array<string> = [];
      childrenResponse.data.children.forEach((child: any)
=> {
        ids.push(child._id);
      });

      let attendanceResponse = await api
        .reports()
        .getChildrenAttendance(ids);

      let dataObject: any = [];
      childrenResponse.data.children.forEach((child: any)
=> {
        child.attendance = attendanceResponse.data.find(
          (report: any) => report.child === child._id
        ).attendance;
        dataObject.push(child);
      });
      setDATA(dataObject);

      getData();
    }, []);

    // ANCHOR get the pressed item and toggle it's attendance in
    DATA
    // and send to the server change the child daily attendace.
  
```



```

    const handleDATAChange = (item: { _id: string; attendance:
boolean }) => {
      let children = DATA.map(
        (child: { _id: string; attendance: boolean }) => {
          if (child._id === item._id) {
            child.attendance = !child.attendance;
            api.reports().updateChildAttendance(
              child._id,
              child.attendance
            );
          }
          return child;
        }
      );
      setDATA(children);
    };

    return (
      <View style={styles.container}>
        <FlatList
          data={DATA}
          numColumns={3}
          columnWrapperStyle={styles.column}
          keyExtractor={(item) => item._id}
          renderItem={({ item }) => (
            <View>
              <TouchableOpacity
                style={styles.item}
                // onPress={() => toggleItem(item)}
                onPress={() =>
handleDATAChange(item)}
              >
                <Image
                  style={
                    item.attendance
                      ? [styles.image,
styles.selected]
                    : styles.image
                  }
                  source={{ uri: item.profilePic }}
                  //
source={require("../assets/images/food.png")}
                />
                <View style={styles.selectedV}>
                  {item.attendance ? Icons.v :
null}
                </View>
                <Text style={styles.name}>
                  {item.name.first}
                {item.name.last}
                </Text>
              </TouchableOpacity>
            </View>
          )}
        </FlatList>
      </View>
    );
  }
}

```

```

const styles = StyleSheet.create({
  container: {
    alignContent: "center",
    marginHorizontal: 30,
    marginTop: 20,
  },
  column: {
    justifyContent: "space-between",
  },
  item: {
    alignItems: "center",
    marginBottom: 25,
  },
  image: {
    borderRadius: 500,
    width: globalStyles.window.width * 0.22,
    height: globalStyles.window.width * 0.22,
  },
  name: {
    fontSize: 12,
    lineHeight: 16,
    alignItems: "center",
    textAlign: "center",
    letterSpacing: 0.1,
    color: globalStyles.color.text,
    fontFamily: globalStyles.font.bold,
    marginTop: 10,
  },
  selected: {
    opacity: 0.6,
    borderWidth: 3,
    borderColor: globalStyles.color.purple,
  },
  selectedV: {
    position: "absolute",
    justifyContent: "center",
    height: globalStyles.window.width * 0.22,
    alignItems: "center",
  },
});

const mapStateToProps = (state: any) => {
  const { user } = state;
  return { user };
};

export default connect(mapStateToProps)(AttendanceScreen);

```

LoginController.js

```
const express = require("express");
const router = express.Router();
const mongoose = require("mongoose");
const { authenticateToken, generateAccessToken } =
  require("../auth/auth");
const ParentsService = require("../services/ParentsService");
const StaffsService = require("../services/StaffsService");

//ANCHOR checks if the user is exist by his email and password
and create a unique token. else, send 401 status.
router.post("/login", async (req, res) => {
  let token;
  if (req.body.data.role === "parent") {
    let user = await
    ParentsService.getParentByEmailAndPassword(
      req.body.data.email,
      req.body.data.password
    );
    ; (
      token = user
      ? generateAccessToken({ id: user._id, role: "parent"
    })
      : null;
    { else if (req.body.data.role === "staff") { ("
      let user = await
      StaffsService.getStaffByEmailAndPassword(
        req.body.data.email,
        req.body.data.password
      );
      ; (
        token = user
        ? generateAccessToken({ id: user._id, role: "staff"
      })
        : null;
      {
        token ? res.send(token) : res.sendStatus(401)
      }
    );
  }

  //ANCHOR getMe request gets the user token to verify it and
  returns data about the user.
  router.get("/getMe", authenticateToken, async (req, res) => {
    if (req.user.role === "parent") {
      const user = await
      ParentsService.getParentById(req.user.id);
      user.role = "parent;";
      res.send(user);
    }
    { else if (req.user.role === "staff") { ("
      const user = await
      StaffsService.getStaffById(req.user.id);
      user.role = "staff;";
      res.send(user);
    }
    { else {
      res.sendStatus(401)
    }
  }
});

module.exports = router;
```

ChildrenService.js

```
const ChildrenRepository =
require("../repositories/ChildrenRepository");
const { Child } = require("../models/ChildModel");
const mongoose = require("mongoose");
const objectId = mongoose.Types.ObjectId;

const getChildNameAndPic = async (id) => {
  return await ChildrenRepository.getNameAndPicById(id);
};

const getChildrenNameAndPic = async (ids) => {
  let childrenList = [];
  for (i = 0; i < ids.length; i++) {
    let child = await
ChildrenRepository.getNameAndPicById(id);
    childrenList.push(child);
  }
  return childrenList;
};

const createNewChild = async (data) => {
  let child = new Child({
    name: {
      first: data.childFirstName,
      last: data.childLastName,
    },
    birthDate: new Date(data.year, data.month, data.day - 1),
    school: new objectId(data.school),
    active: true,
  });
  child = await child.save();
};

module.exports = { getChildNameAndPic, createNewChild,
getChildrenNameAndPic };
```

ReportRepository.js

```
const mongoose = require("mongoose");
const { Report } = require("../models/ReportModel");
const endOfDay = require("date-fns/endOfDay");
const startOfDay = require("date-fns/startOfDay");

const getChildrenAttendanceByChildrenIds = async (ids) => {
  return await Report.find(
    {
      child: { $in: ids },
      date: { $gte: startOfDay(new Date()), $lte:
endOfDay(new Date()) },
    },
    "attendance child"
  );
};

const updateAttendanceByChildId = async (id, attendance) => {
  return await Report.findOneAndUpdate(
    {
      child: id,
      date: { $gte: startOfDay(new Date()), $lte:
endOfDay(new Date()) },
    },
    { attendance: attendance }
  );
};

const addSubReportToReportByChildId = async (id, subReport) => {
  return await Report.updateOne(
    {
      child: { $in: id },
      date: { $gte: startOfDay(new Date()), $lte:
endOfDay(new Date()) },
    },
    { $push: { subReports: subReport } }
  ); // TODO not sure this is the right way to do that
};

module.exports = {
  getChildrenAttendanceByChildrenIds,
  updateAttendanceByChildId,
  addSubReportToReportByChildId,
};
```