Expecto patronum

אימוץ ילדים מרחוק

מגיש: עלי נאבסו  
302946371  
מנחה: איאד סולימאן  
מכללת תל חי

תוכן עניינים

[תקציר 2](#_Toc105861349)

[Abstract 2](#_Toc105861350)

[רשימת סמלים וקיצורים 3](#_Toc105861351)

[פרק מבוא 4](#_Toc105861352)

[הבעיה 4](#_Toc105861353)

[אופן הפתרון 4](#_Toc105861354)

[תיאור פעולת האתר 5](#_Toc105861355)

[אופן פעולת האתר 5](#_Toc105861356)

[מבנה התוכנה 11](#_Toc105861357)

[הטכנולוגיות 11](#_Toc105861358)

[למה בחרתי בספריות האלו? 11](#_Toc105861359)

[למה להפריד לשני פרויקטים? 12](#_Toc105861360)

[המודלים ומבנה המסד נתונים 13](#_Toc105861361)

[אופן פעולת הAPI 14](#_Toc105861362)

[אופן פעולת הUI 15](#_Toc105861363)

[סיכום ומסקנות 16](#_Toc105861364)

[תכנון מוקדם חוסך זמן 16](#_Toc105861365)

[לפצל את הקוד איפה שצריך ולבנות components איפה שצריך 16](#_Toc105861366)

[לא הכול חייב להיות אוניברסלי 17](#_Toc105861367)

# תקציר

בעולם של היום, לצערנו, מלחמות שכיחות בכל רכבי העולם וזהו המציאות היומית של מליוני אנשים ברכבי העולם. ואחרי שכל ההרס והחורבן נגמרים ונשכחים על ידי העולם, אינספור יתומים נשארים לבדם.

האתר מיועד לעמותה שכל פועלה היא לדאוג לכל אותם יתומים – במיוחד ליתומים שהיגרו למדינות שכנות ואין מי שידאג להם.

האתר מאפשר לעמותה בערתם האדיבה של אנשים מכל העולם לתרום כסף ולממן את הצרכים הבסיסים של אותם יתומים

# Abstract

In today’s world, war and destruction are common and frequent and that is the day-to-day norm for millions of peoples and after the dust settles, countless children are left orphans to fend for their self.

The website was built for a nonprofit organization that it main goal is to provide for those children using a donation from people all over the world that choose to sponsor some of those kids.

# פרק מבוא

## הבעיה

בעולם של היום, לצערנו, מלחמות קורות כל יום ויום. ואחרי שכל ההרס והחורבן נגמרים ונשכחים על ידי העולם, אינספור יתומים נשארים לבדם.

האתר שבניתי מיועד לעמותה שכל פועלה היא לדאוג לכל אותם יתומים – במיוחד ליתומים שהיגרו למדינות שכנות ואין מי שידאג להם.

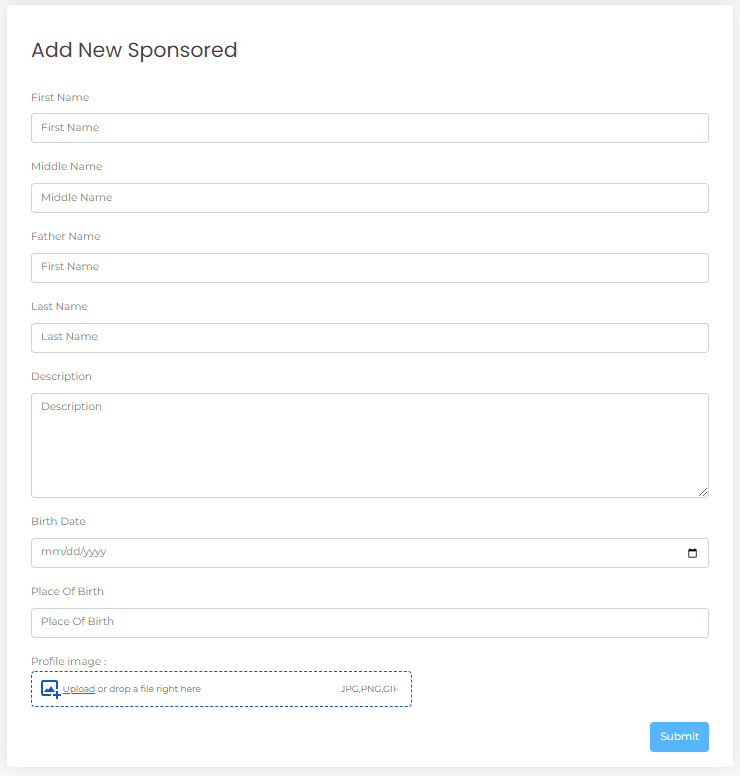
## אופן הפתרון

האתר מאפשר לאנשים מכל העולם לתרום כסף ולממן את היתומים ביום יום או עד יעד מסוים.

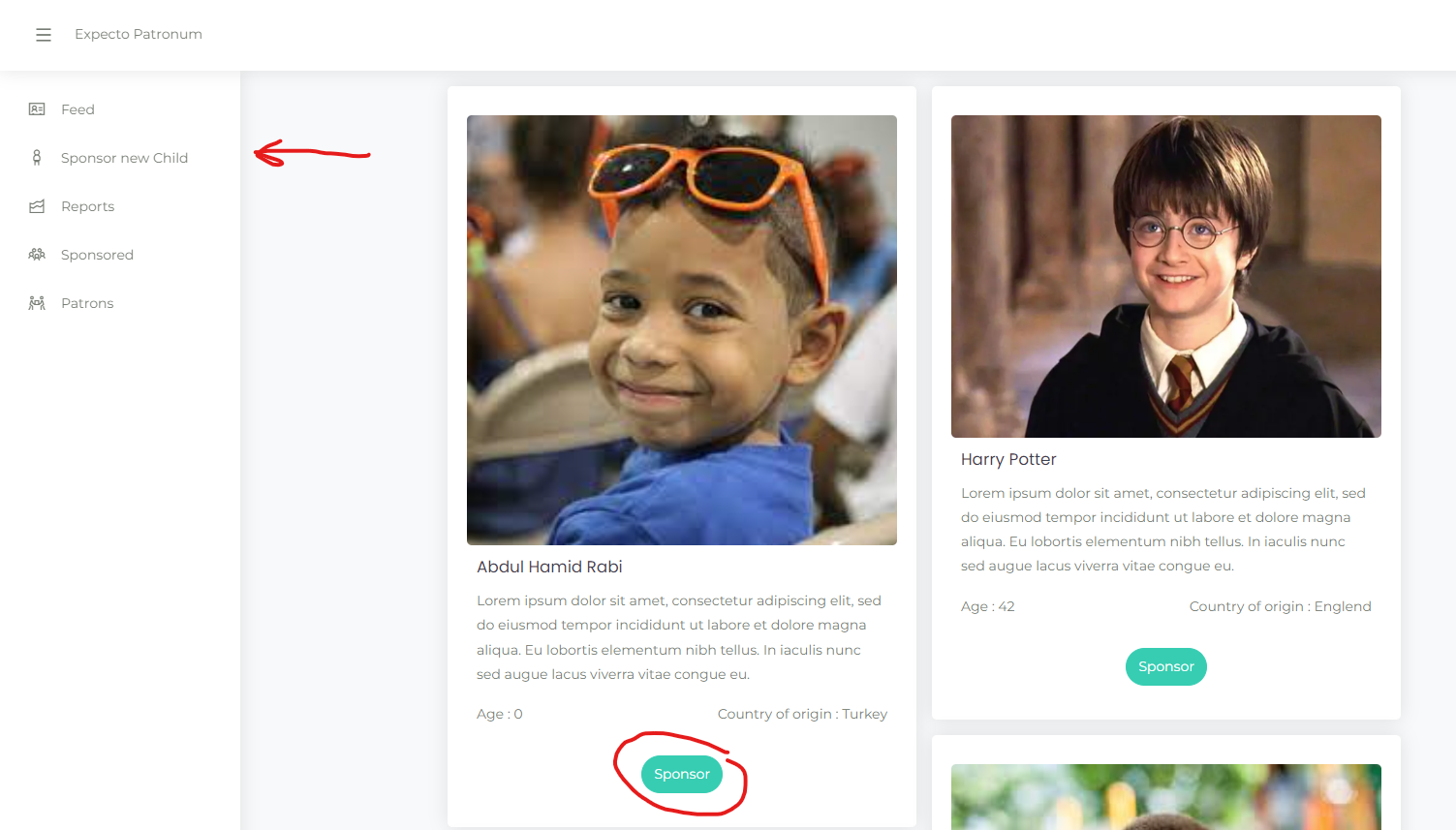
כל משתמש שנכנס לאתר יכול להיכנס לגלריה, לבחור את היתום אותו הוא רוצה לממן ולאיזה תקופת זמן.

# תיאור פעולת האתר

## אופן פעולת האתר

תחילה, העמותה מכניסה את הנתונים של הילדים ומעלה תמונות וכל מידע רלוונטי שיכול לתאר את הילד ככה שהתורמים יוכלו להתחבר לילד.  


התורמים נרשמים לאתר ועוברים על הגלריה של הילדים ובוחרים את מי הם רוצים לממן ולכמה זמן (חד פעמי, שישה חודשים, שנה או עד סיום בית הספר).



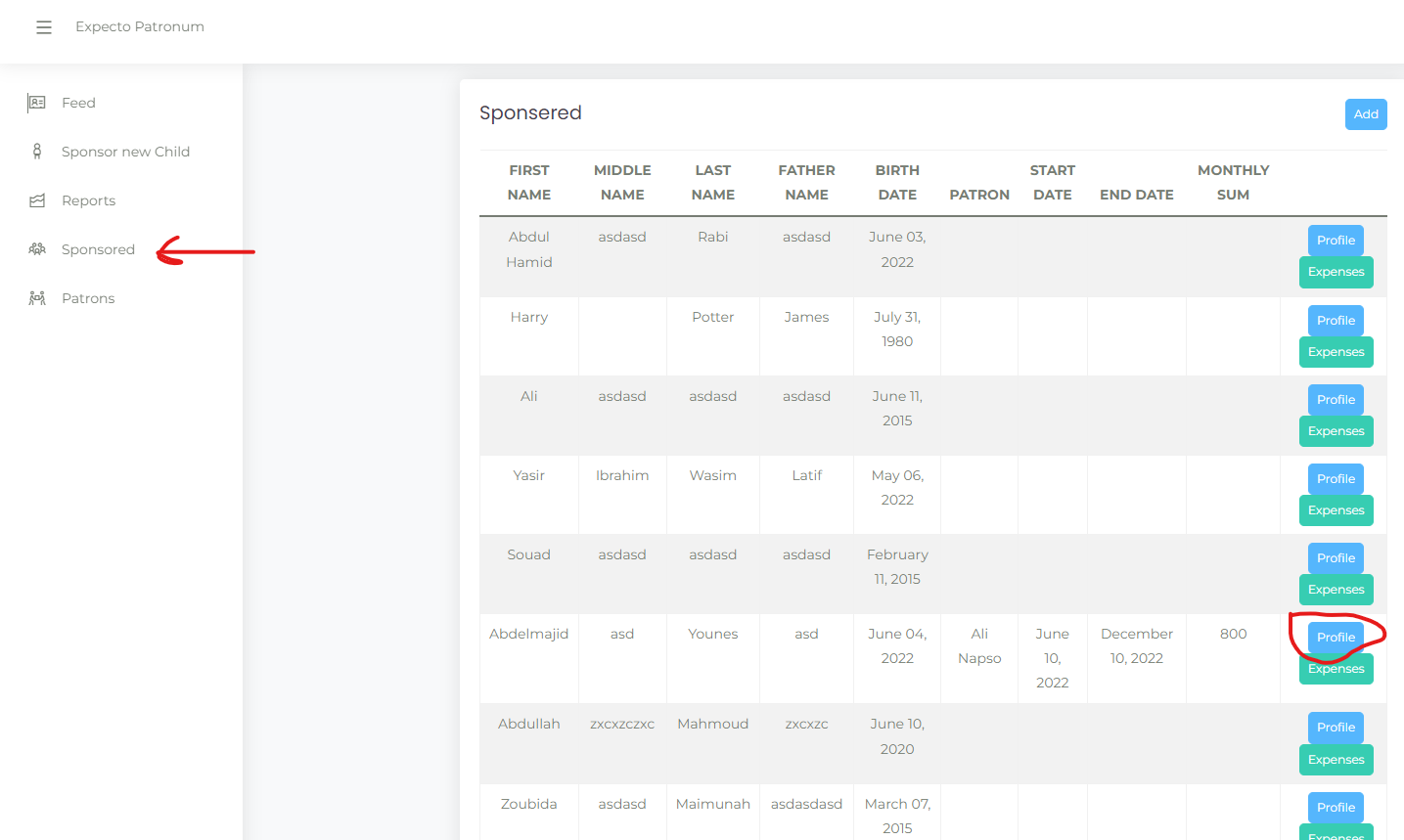
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

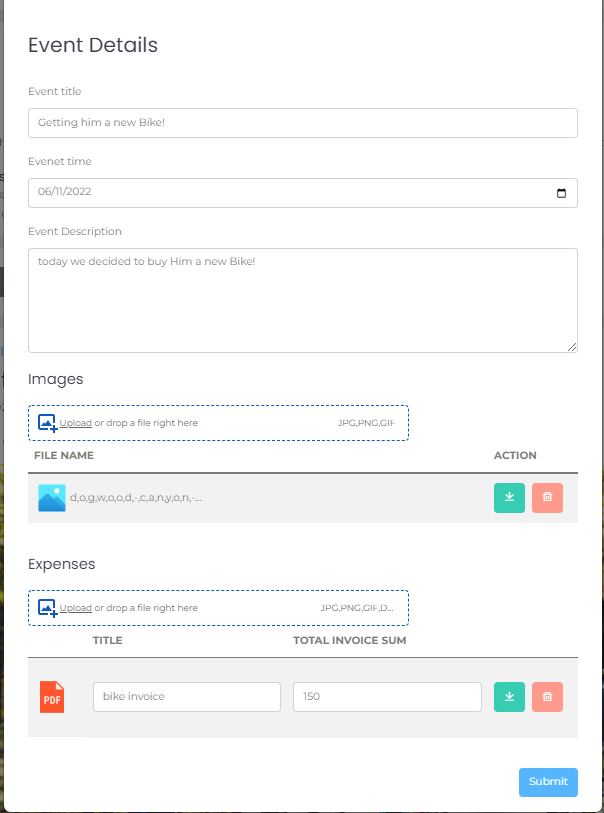
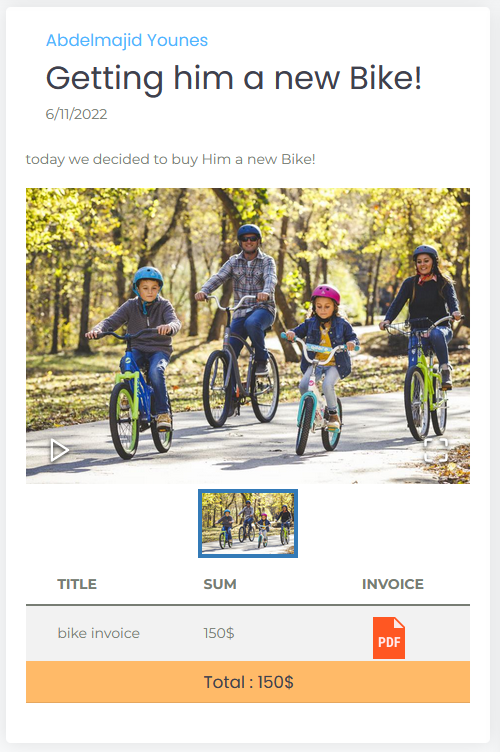
לאחר הסדרת התשלום, מנהלי העמותה יחליטו מה הילד צריך להתנהלות היום יומית שלו (אוכל, ביגוד, צעצועים...) והם ירכשו את מה שנדרש

לאחר שכל מה שצריך נרכש יעלו המנהלים את התמונות והחשבוניות לאתר עם תיאור של מה נעשה באותה רכישה, המנהלים לא צריכים ל"בזבז" את כל הכסף בפעם אחת ויכולים לפרוס את הרכישות במהלך החודש.

המנהלים יצרו פוסט חדש מדף הפרופיל של הילד –

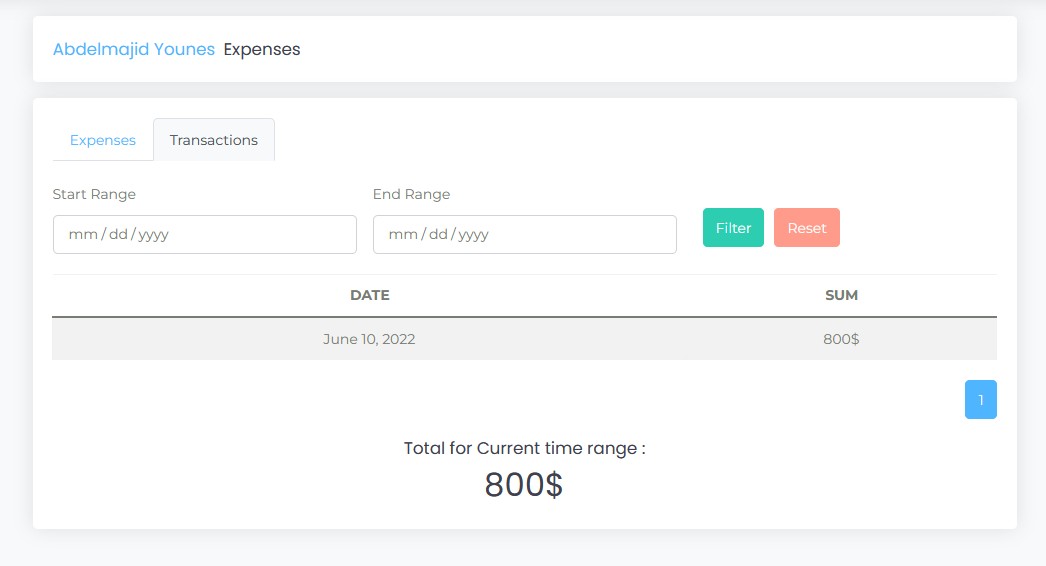
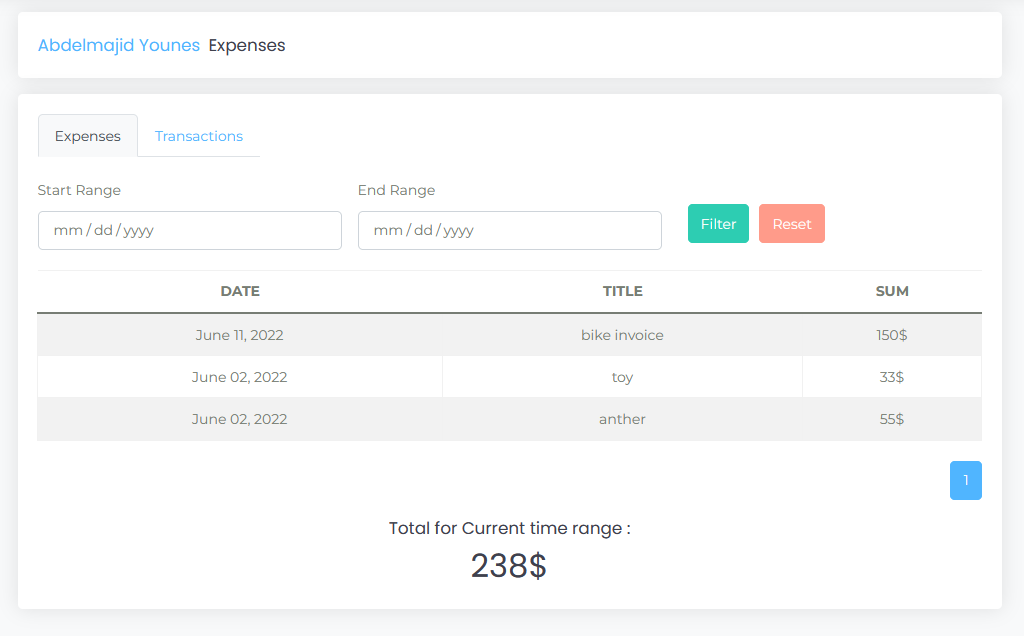
  
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

לאחר שהמנהלים יעלו את התמונות והחשבוניות, יוכל התרום לראות את התיעוד באתר עם פירוש של סכום של כל חשבונית וסה"כ של החשבוניות  


תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיהמשתמש יוכל לסנן את החשבוניות לפי פרקי זמן ולראות את החיובים שנעשו על ידי האתר המנהלים יוכלו לראות מי ממומן על ידי מי ומי לא חיובים שנעשו וכו.



# מבנה התוכנה

## הטכנולוגיות

הפרויקט מחולק ל4 חלקים:

1. Frontend – נכתב בעזרת [nextJS](https://nextjs.org/), ספריית פרונטאנד מבוססת react
2. Backend- נכתב בעזרת [nestJS](https://nestjs.com/) ספרייה לכתיבת API [וPrisma](https://www.prisma.io/) ספרייה להתחברות לכל מסד נתונים ולהתנהל באופן אבסטרקטי מול כל מסד נתונים.
3. Common – ספרייה פרטית שמקשרת בין שני הפרויקטים שמכילה את כל המודלים של האובייקטים
4. מסד נתונים של [postgres](https://www.postgresql.org/) לשמירה של הנתונים

## למה בחרתי בספריות האלו?

* **NextJs**: זה לא סוד שכיום React היא הספרייה השולטת בכל מה שנוגע לUI ופרונטאנד אבל להשתמש בריאקט לבד ולנהל את הכול לבד זה תהליך ארוך מייגע ומסובך ו nextJS היא ספרייה שבאה לפתור את כל החלקים שכל מתכנת הולך לעשות בכל אתר ומנהלת אותם בעצמה, וכל מה שאת\ה צריך לעשות זה לנהל את הדפים ככל דף ולדאוג רק לאיך הדאטה יגיע לדף ולא לדאוג איך להגיע לדף.  
  לדוגמא: כדי ליצור דף חדש עם נתיב ספציפי, כל מה שעליך לעשות זה ליצור תיקייה חדשה בתיקייה של הדפים ולקרוא לה בשם שאתה רוצה.
* **nestJS**: בעזרת הספרייה הנ"ל, המפתח יכול להחליט איך הAPI שלו יראה ואיך הוא ייחשף די בקלות, בנייה אוטומטית של התיעוד ולדאוג לענייני אבטחה בקלות יחסית ולבחור איך להתחבר למסד הנתונים.  
  בעזרת Prisma נוכל להתחבר לכל מסד נתונים שנרצה, בין אם זה טבלאי או אחר ופריזמה ידאג להכול. כל מה שהמתכנת צריך לדאוג לו זה איך המודלים יראו ופריזמה תדאג (בהגדרת קובץ מתאים) לבנות את כל הטבלאות והקישורים המתאימים במסד הנתונים ולעדכן את העמודות במידה ושינוי עתידי יצרך.

בשתי הספריות ניתן לכתוב בTypescript – מה שהקל על שתכנות, ומה שאפשר לבנות את הספרייה הכללית שמשתפת את הקוד בין שני הפרויקטים

## למה להפריד לשני פרויקטים?

* הסיבה הראשונה הייתה לימוד, רציתי לראות איך זה לעבוד עם API מרוחק, ואיך זה לבנות API שעומד על עצמו ואפשר להשתמש בו לבד.
* הסיבה השנייה ואולי היותר חשובה היא יצירת הפרדה בין הFRONTEND והBACKEND לצורכי אבטחה, ולהבין מתי המשתמשים כן יכולים לגשת למידע ומתי לא באופן אינטואיטיבי. הדבר מקל גם על ניתור כמות הקריאות למסד ולצמצם אותם.
* במידה ונרצה להוסיף שירותים נוספים, כמו אפליקציה לפלאפון. בAPI לא נצטרך לשנות דבר ורק לצרוך את הAPI איך שהוא.
* מקל על הפיתוח – אם יש באג בUI, לא צריך לבנות מחדש את הAPI.

# המודלים ומבנה המסד נתונים

|  |  |
| --- | --- |
| model User {    id           String    createdAt    DateTime    updatedAt    DateTime    email        String    hash         String    firstName    String?    lastName     String?    Address      String?    role         Role?    isActive     Boolean  }  enum Role {    PATRON    ADMIN  }  model Sponsored {    id               String            @id    createdAt        DateTime    updatedAt        DateTime    firstName        String    middleName       String    fatherName       String    lastName         String    birthDate        DateTime    placeOfBirth     String    description      String    uploadedFileId   String?    isActive         Boolean    patronId         String?    monthlyDum       Float?    startDate        DateTime?    endDate          DateTime?    dayOfTransaction Int?  } | model UploadedFile {    id                String           @id    title             String    postfix           String    fileCategory      FileCategory    sponsoredEventsId String?  }  enum FileCategory {    PROFILE    IMAGE    DOC  }  model SponsoredEvents {    id          String         @id    createdAt   DateTime    updatedAt   DateTime    eventDate   DateTime    title       String    description String    sponsoredId String  }  model Transactions {    id          String    @id    sum         Float    patronId    String    sponsoredId String    createdAt   DateTime    updatedAt   DateTime  }  model Expenses {    id               String           @id    sum              Float    title            String    sponsoredEventId String?    uploadedFileId   String?  } |

# אופן פעולת הAPI

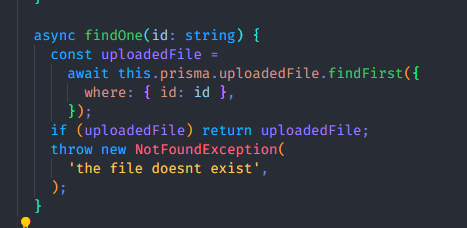
לAPI יכולים לגשת שלושה סוגים של משתמשים

1. לא רשומים – יכולים רק להירשם,
2. רשומים – יכולים לראות פרטים רק על עצמם וילדים שהם ממנים
3. מנהלים – יכולים לעשות הכול

הAPI בעזרת תגים מנהל את ההרשאות של כל endpoint וגם בנוסף ישנם נקודות שבהן גם רשומים וגם מנהלים יכולים לגשת, שם הAPI יודע לזהות את סוג המשתמש ולסנן את התוצאות בהתאם.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטילדוגמא – לendpoint להוספת ילד חדש יש תג של ADMIN ורק מנהל יכול להוסיף ילד למערכת

הAPI מתחבר למסד נתונים באמצעות ספריית PRISMA – ספרייה שמנהלת את כל ההתחברות למסד

# אופן פעולת הUI

בדומה לAPI גם לUI יש 3 רבדים של אבטחה

את האבטחה מנהל context כללי שרץ על כל האתר, הוא יודע לזהות אם מחובר משתמש, מה ההרשאות שלו והאם הוא אמור להיות בדף הספיציפי. (נמצא ב /src/context/user.tsx)

את כל הקריאות לAPI מנהלת פונקציה אחת שיודעת לפי סוג הקריאה לבנות את הקריאה ולהחזיר אותה מהAPI לUI (lib\remoteAPI.ts)

היות ויש הרבה טבלאות באתר, נוצרו components מיוחדות לשימוש חוזר איפה שצריך.

דוגמה טובה לרכיב כזה היא הטבלה.

כמעט בכל הדפים ישנה טבלה, לכן הסדרה של איך הAPI מחזיר את התוצאות, אפשרה לעשות טבלה אוניברסלית לכל מקום שהיה צריך

כל מה שצריך לספק לטבלה היא מאיזה endpoint אנחנו רוצים את המידע ,איזה עמודות אתה רוצה לראות, מה הכותרת של העמודה ואיך היא ממופה בתוצאות שחוזרות. אפשר גם להגדיר עיצוב מיוחד לתוכן של התא אם רוצים.

דוגמא למיפוי  
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

דוגמא לשימוש  


# סיכום ומסקנות

## תכנון מוקדם חוסך זמן

בזמן הפיתוח היו מקרים בהן דרך הפעולה נראית ברורה ופשוטה, אבל היא לא. עצירה ומחשבה על מה אני עושה, מה יהיו ההשלכות להחלטה הזאת חוסכים בטווח הרחוק יותר זמן.

לדוגמא: בזמן פיתוח הטבלה, שהיא ניתנת למיון לפי כל עמודה, בסדר עולה או יורד, מיון העמודות נעשה בצד לקוח.

למה זה היה טעות? לאחר שהטבלה עברה את ה15 שורות והטבלה התחלקה לדפים.

הAPI מחזיר רק את ה15 הראשונים ומיון של אותם 15 לא אמור בהכרח אלה הן השורות שצריכות להופיע. לכן היה צריך קודם למיין בAPI ואז להחזיר לצד לקוח את המידע כבר ממיון

## לפצל את הקוד איפה שצריך ולבנות components איפה שצריך

שימוש חוזר בקוד חוסך זמן בכתיבה ובדיבוג. במקומות שעושים העתק הדבק כי זה כולה עוד כמה שורות גורמות לבעיות לאחר מכן.

לדוגמא:

למשתמש יש שני תאבים שמאפשרים לו לראות שתי טבלאות. אופן הפעולה של שתי הטבלאות זהה למעט מאיפה הוא לוקח את המידע.

מה שייחודי לשני התאבים (ועד אז רק להם) היא האפשרות לסנן לפי תאריך.

היום ורק שתי התאבים מנצלים את הפונקציונליות הזאת הועתקו חלקים זהים בקוד

במקום לבנות קומפוננטה

דיבוג של שעתיים העלה כי שתי הטבלאות השתמשו באותו הstate מה שגרם להתנהגויות מוזרות.

## לא הכול חייב להיות אוניברסלי

אם האוניברסליות מתחילה לסבך את הקומפוננטה, זה לא נורא לפצל ל2

לדגומא:

ישנה קומפוננטה שאחראית על העלאת הקבצים לשרת.

אותה קומפוננטה יכול להיות בשימוש בשלושה אופנים

1. לתמונת פרופיל
2. לתמונות בפוסט
3. מסמכים (חשבוניות)

בכל אחת מאופני השימוש הקומפוננטה מתנהגת אחרת

1. תמונת פרופיל יכולה לקבל רק תמונה, ורק אחת
2. תמונות בפוסט יכולים להיות כמה שרוצים, השם של הקובץ לא משנה, ולא צריך תיאור
3. המסמכים יכולים להיות מפורמטים שונים, וגם אנחנו רוצים לשמור בנוסף את סכום החשבונית ותיאור קטן

בניה של קומפוננטה אחת שמספקת את כל הדרישות של כל אופני השימוש התגלתה כמסובכת גדולה וקשה לדיבוג לכן פוצלה ל3 קומפוננטות שונות.

# נספחים

אופן הרצת הפרוייקט

1. API  
   1. yarn install  
   2. yarn start:dev
2. Frontend  
   1. yarn install  
   2. yarn dev

המסד נתונים מקונפג ורץ בשרת