***EasyTravel* – מערכת המלצות לתכנון טיולים**

**כיצד להריץ את הפרויקט**

קישור ל-*Git*: <https://github.com/tehilooli/EasyTravel>

דרישות המערכת

1. *NPM* v6.13.4
2. *Node.js* v12.16.0
3. דפדפן
4. סביבה מתאימה להרצת קובץ *Python* להקמת השרת, כגון *PyCharm*
5. סביבה מתאימה להרצת פקודות בשפת *SQL* ליצירת ה-*DB* – *MySQL*

הרצת הפרויקט

*עבור ה-DB:*

לפתוח *Query* חדש בתוכנה *MySQL*, ושם להריץ יחד את כל הפקודות בקובץ *easytravel\_db.sql*.

*עבור ה-Back End:*

נפתח את הקובץ *main.*py בסביבה מתאימה להרצת קובץ *Python*, ונריץ את ה-*command lines* הבאות:

1. *pip install Flask*
2. *pip install flask-cors –target*
3. *pip install flask-mysqldb*
4. *set FLASK\_APP=main.py*
5. *flask run*

*עבור ה-Front End:*

נריץ את ה-*command lines* הבאות:

1. *git clone*
2. *cd easy\_travel*
3. *npm install*
4. *npm start*

**מבוא**

היקף

בפרויקט המטרה שלי היא ליצור אתר המרכז למשתמש את הדברים הנצרכים לו לתכנון טיול. הרעיון הוא שכלל הדברים הנחוצים יהיו תחת פלטפורמה אחת. למשל: חיפוש טיסות, חיפוש מקומות לינה, המלצות לאטרקציות ומסעדות, ועוד. מטרה נוספת היא להנגיש למשתמש את המידע והשימוש באתר בצורה כמה שיותר נוחה וקלה לתפעול.

סקירה

אציג את המסכים השונים באתר, את רכיבי המערכת השונים וכן אתן הסבר קצר על כל אחד מהרכיבים.

טרמינולוגיה

1. ***React* –** ספריית קוד פתוח בשפת *JavaScript* המשמשת כתשתית לפיתוח ממשקי משתמש.
2. ***Component* –** אובייקט לוגי שניתן שיהיה לו ייצוג על המסך.
3. ***Props* –** ארגומנטים המועברים ל-*component* כלשהו על ידי ה-*component* שמייצר אותם.
4. ***State* –** אובייקט המוגדר בבנאי של ה-*component*, והוא מכיל משתנים המייצגים את המצב הנוכחי של ה-*component*. כלומר, מדובר במשתנים לשימוש פנימי של ה-*component* הנוכחי.   
   לדוגמא, נבנה *component* המייצג כפתור, ונגדיר לו *state* התחלתי: *color: blue*.  
   באתחול התוכנית הכפתור יהיה בצבע כחול.

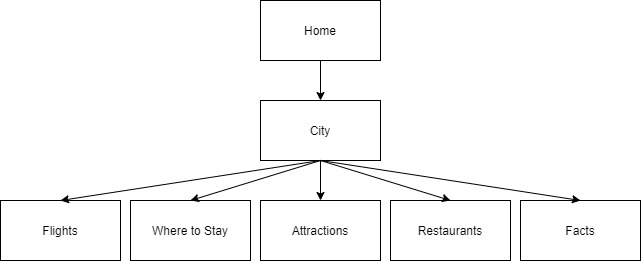
נגדיר שכאשר אנו לוחצים על הכפתור, המשתנה *color* ב-*state* ישתנה ל-*red*. אז בלחיצה על הכפתור, הוא יהפוך לאדום.

**תיאור עיצוב התוכנה**

תזרים התהליך

בתחילה ניתן לבחור מבין רשימת ערים את העיר אליה רוצים לטייל, ולאחר מכן עבור כל אחת מהערים אפשר לבחור בחיפוש טיסות, מקומות לינה ובצפייה באטרקציות ובמסעדות מומלצות.

ארכיטקטורת התוכנה

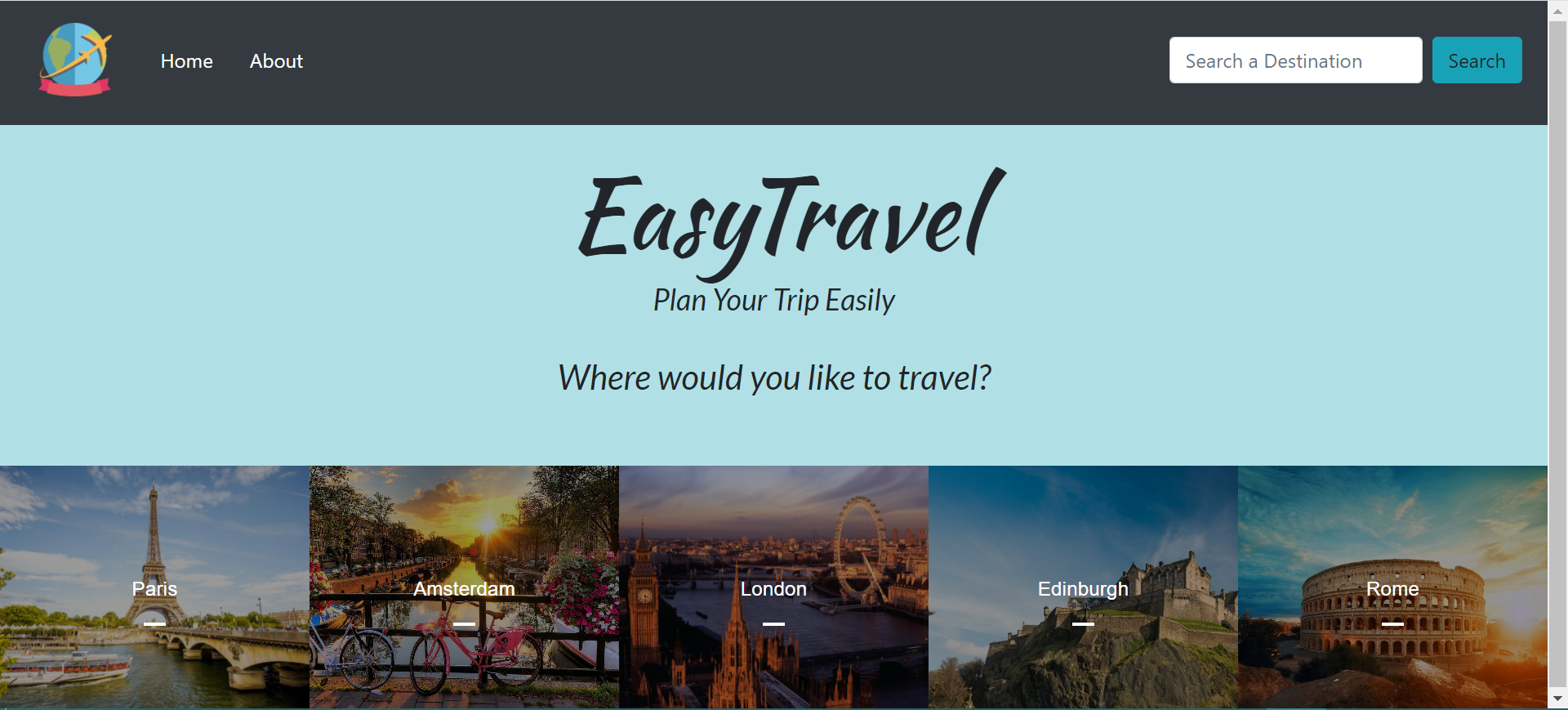
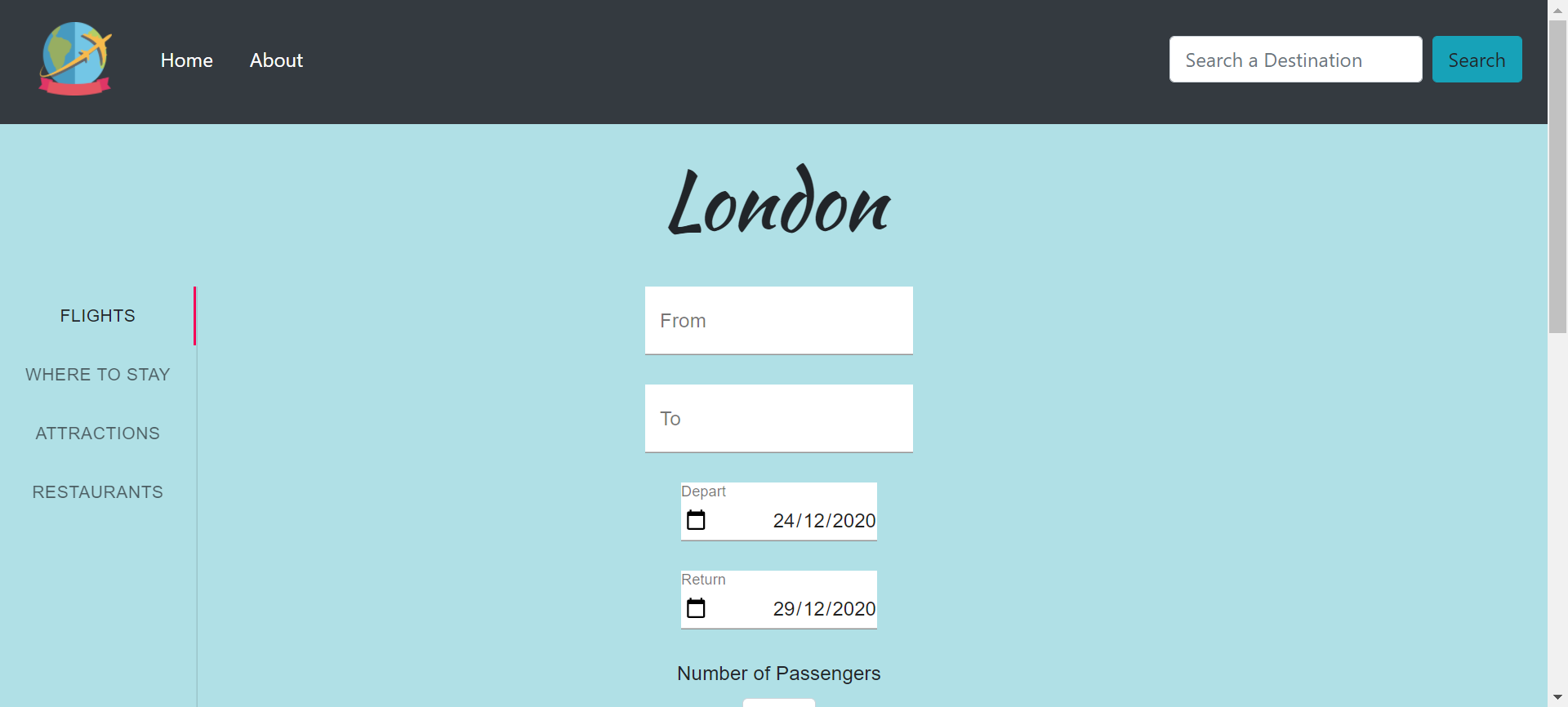
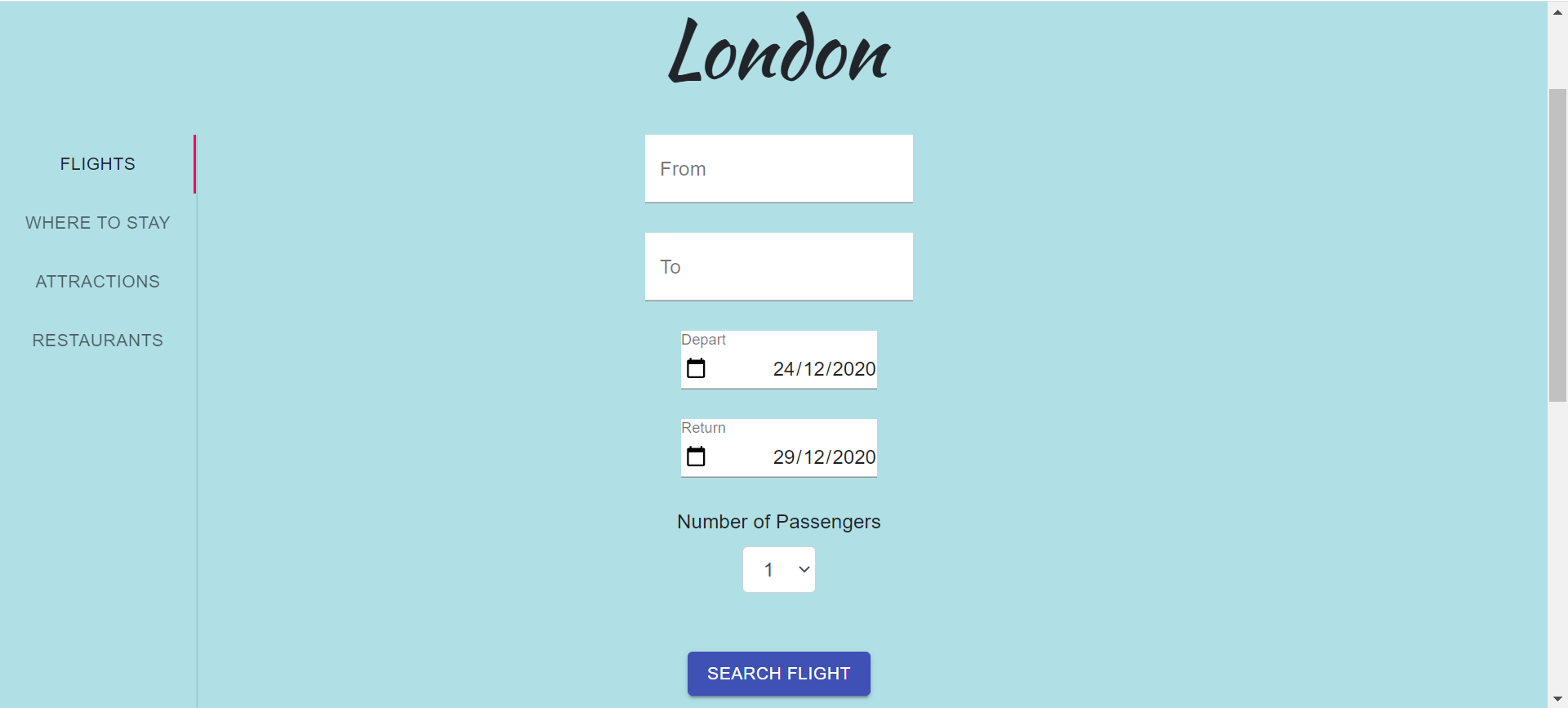
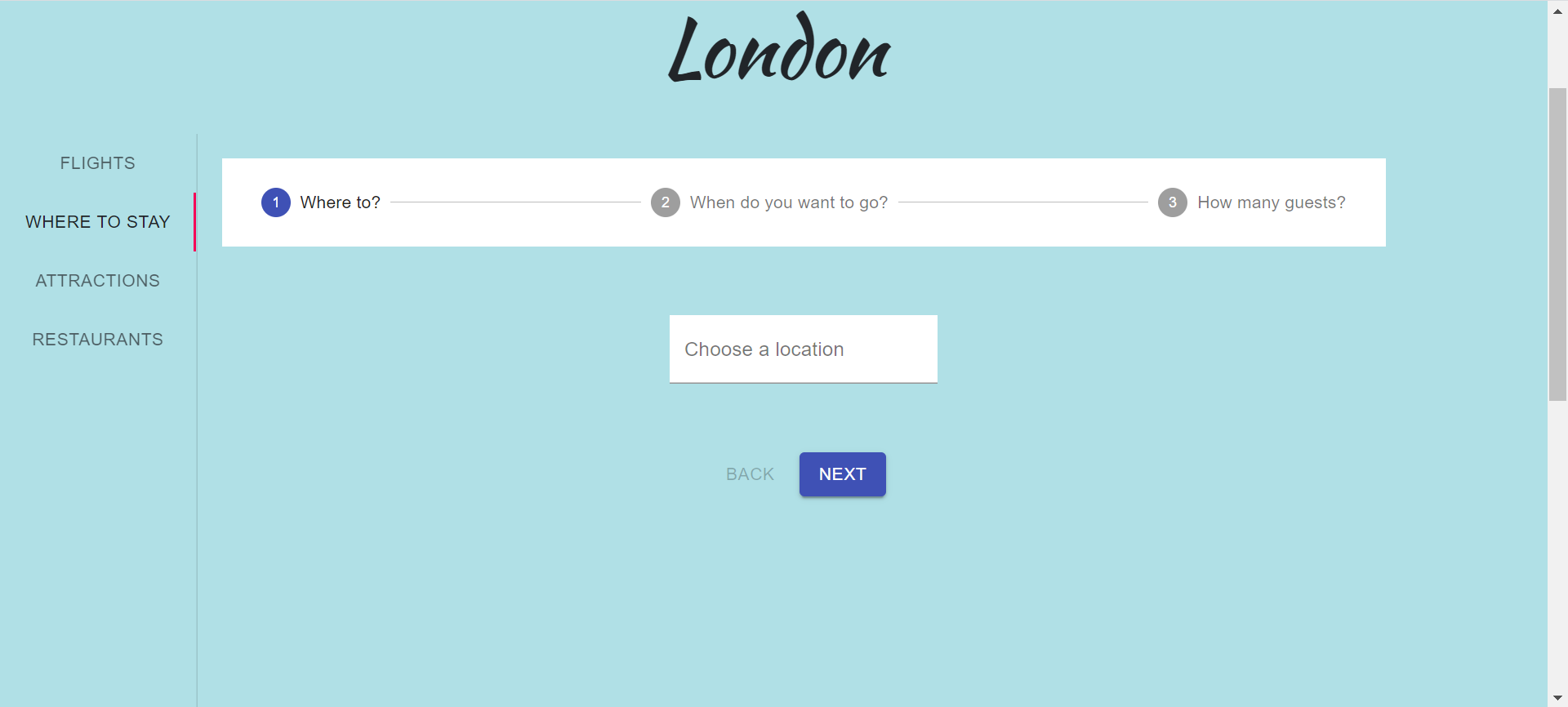


בצד הלקוח, המערכת נוצרת בעזרת *React*, והיא מורכבת מהיררכיה של *components*.

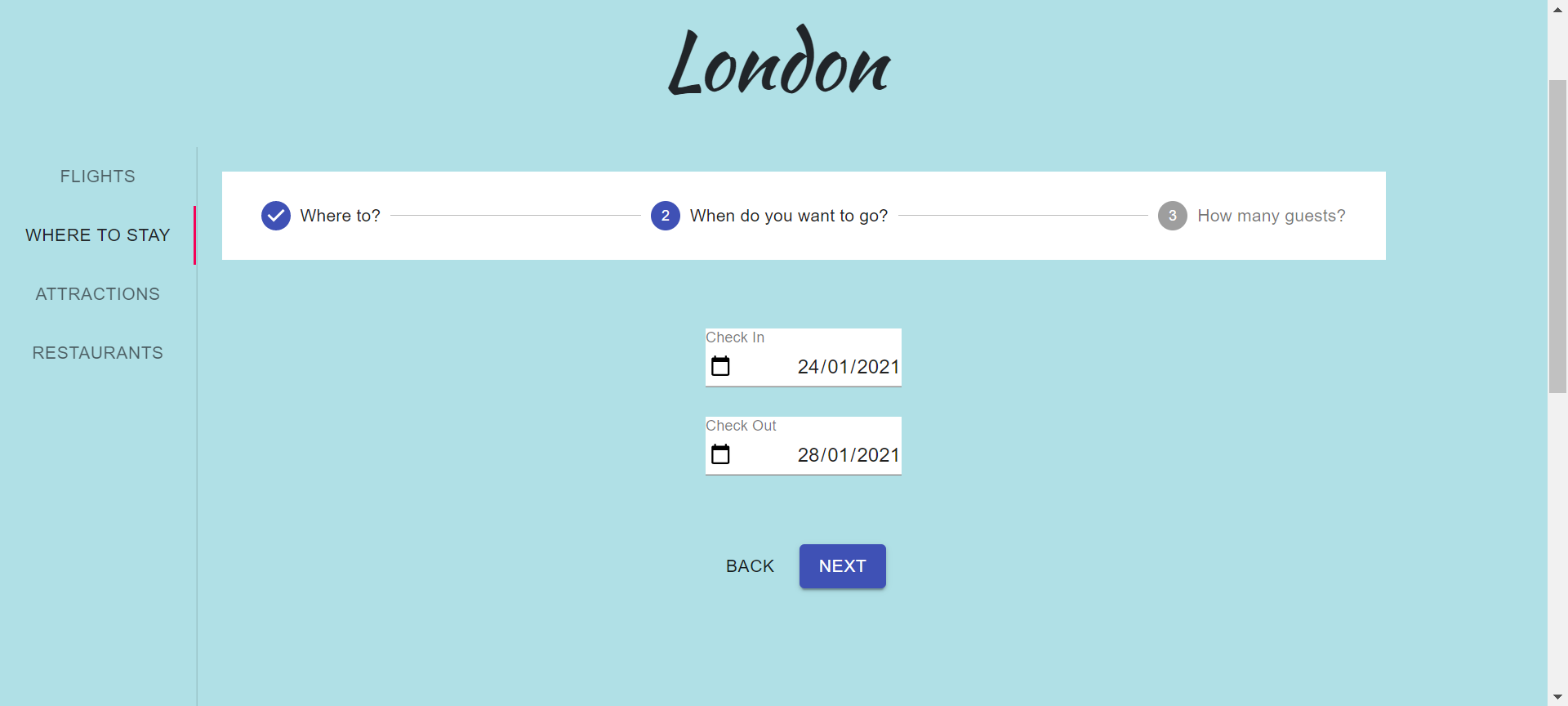
צד השרת הוקם בעזרת סביבת הפיתוח flask בשפת Python.

הנתונים השונים באתר נשלפים בעזרת קריאה מ-DB, בשפת SQL.

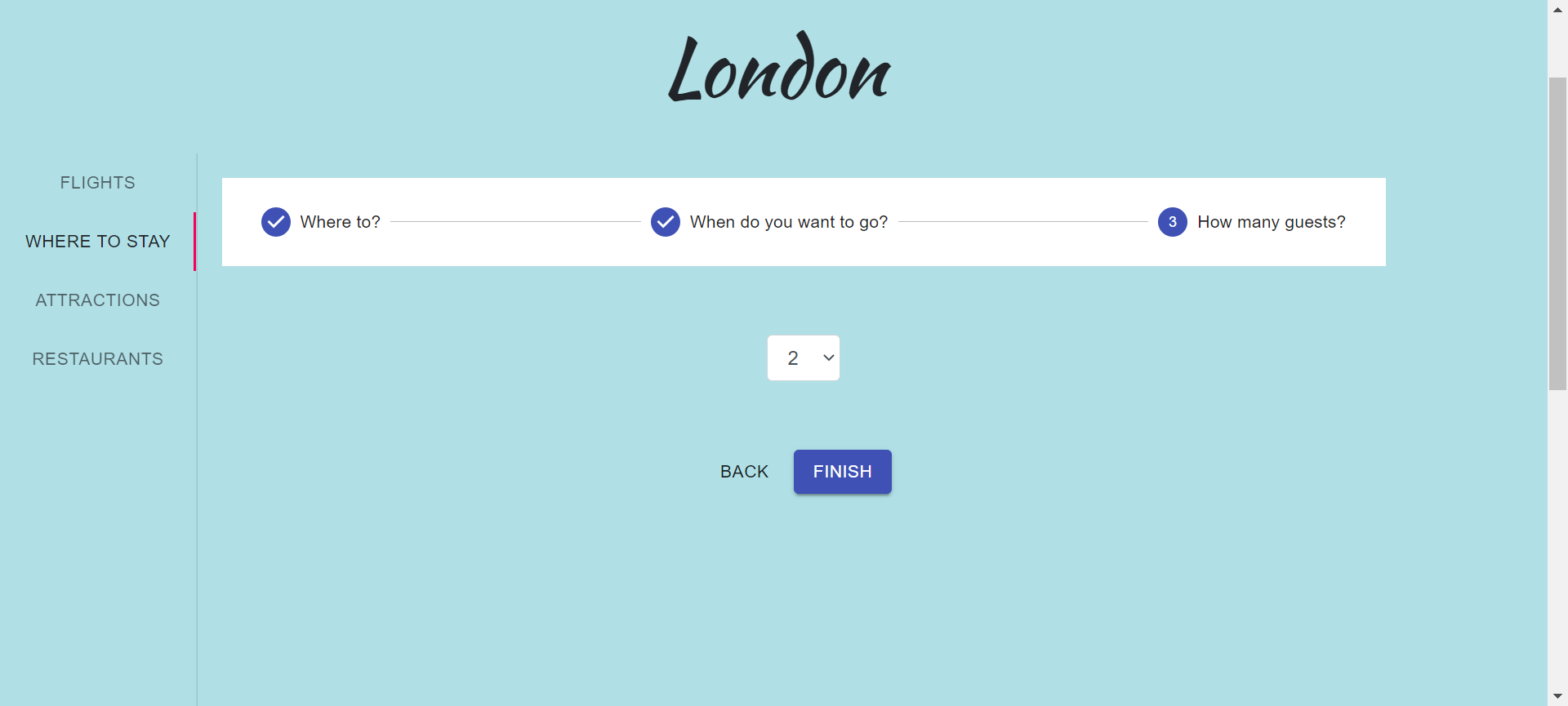
ה-*components* המרכזיים במערכת:

1. ***HomePage* –** מייצג בפועל את הנראות של המסך הראשי, דף הבית. מכיל *component* מסוג *ImageButtonsList*, המייצג את הכפתורים המובילים לערים השונות, כאשר כל כפתור מעוצב בצורת תמונה. את רשימת הערים ונתונים נוספים הקשורים לכל עיר אנחנו שולפים בעזרת השרת מה-DB, ומעבירים אותה בצורת מערך דרך ה-*props* ל-*component* מסוג *ImageButtonsList*.  
     
   
2. ***CityPage* –** מייצג את המסך של כל עיר. מכיל *component* מסוג *VerticalTabs*, המייצג *tabs* הנמצאים בצד שמאל של המסך (כמו כרטיסיות בדפדפן רק בצורה אנכית), בעזרתם יכול המשתמש לעבור בין האופציות השונות הקיימות לכל עיר: טיסות, מקומות לינה, אטרקציות ומסעדות.  
     
   
3. ***Flights* –** יוצר בעזרת *form* שדות להכנסת קלט על ידי המשתמש: מהיכן הוא רוצה לטוס, לאיזה יעד, מהו תאריך היציאה ומהו תאריך החזרה, וכן מהו מספר הנוסעים.  
     
   
4. ***WhereToStay* –** מכיל *component* מסוג *Stepper* אשר מציג למשתמש שדות אותם הוא צריך למלא בנוגע ליעד אליו הוא רוצה להגיע, שלב אחר שלב. בשלב הראשון הוא בוחר את המיקום אליו הוא רוצה להגיע, בשלב השני הוא בוחר את תאריך הצ'ק אין ואת תאריך הצ'ק אאוט, ובשלב השלישי הוא בוחר את מספר האורחים. לבסוף מוצגים לו כל הפרטים שבחר בכל השלבים, ואם הוא רוצה לשנות אותם, ניתן ללחוץ על reset ולבחור מחדש את הפרטים הרצויים.  
   שלב 1: 

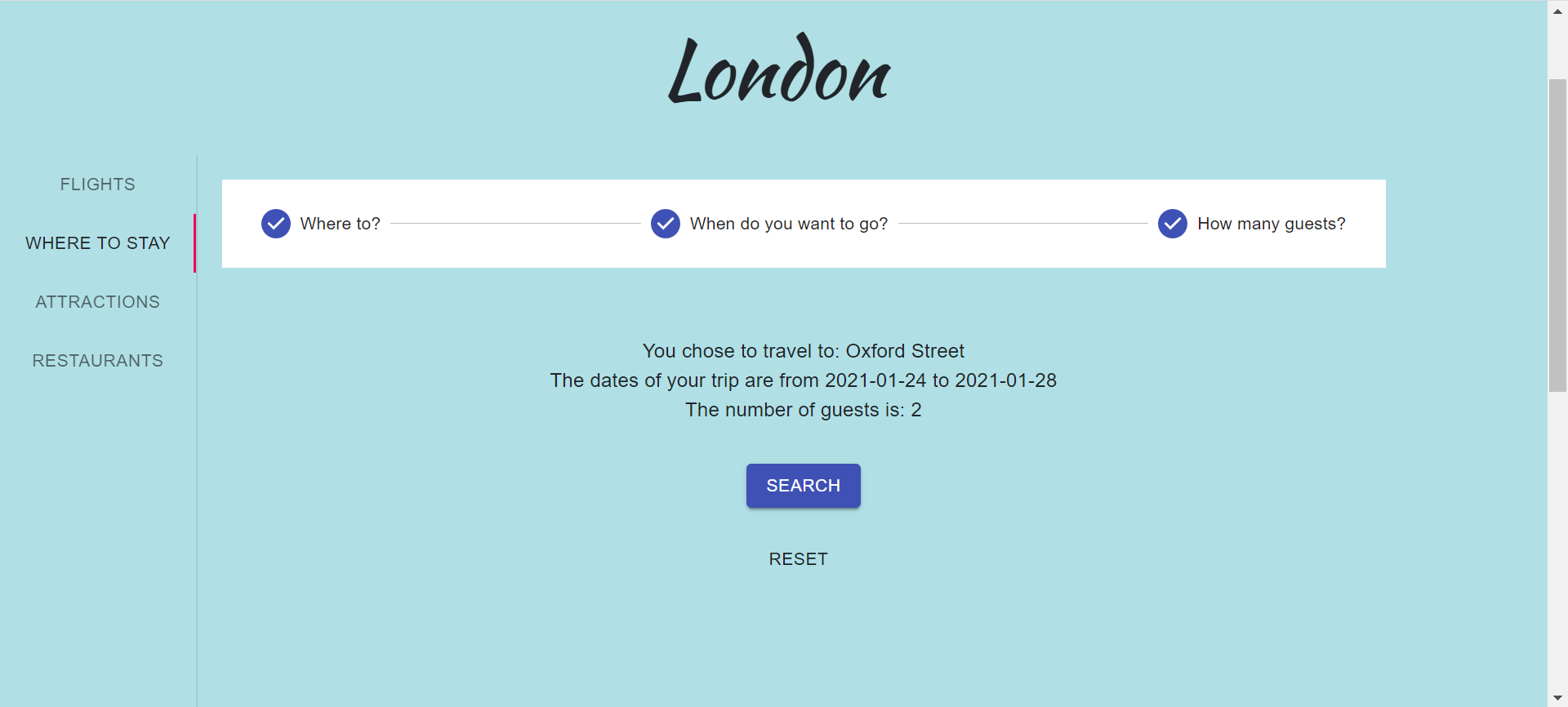
שלב 2:



שלב 3:



שלב סופי:

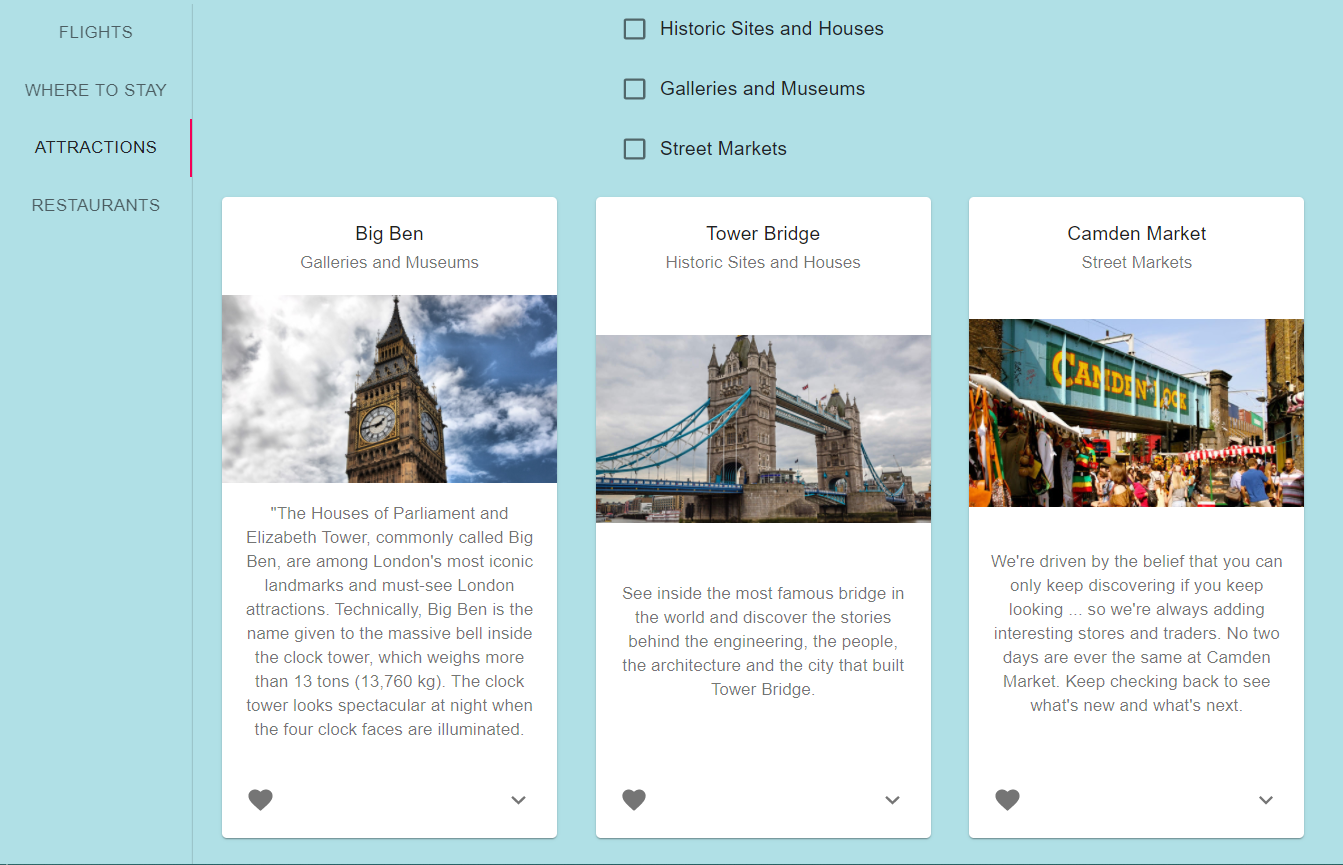


1. ***Attractions* –** ראשית, הוא מכיל *component* מסוג *CardsGrid*, המקבל בעזרת *props* מערך של אטרקציות הנשלף בעזרת השרת (כאשר כל אטרקציה היא אובייקט המכיל שם, כותרת משנה, תמונה, אינפורמציה ראשונית, אינפורמציה נוספת וקטגוריות), ולפיו הוא מוסיף כרטיסיות המייצגות את האטרקציות הללו – בעזרת *component* מסוג *Card*.

בנוסף, הוא מאפשר לסנן את האטרקציות לפי קטגוריות שונות:

1. *Historic Sites and Houses*
2. *Galleries and Museums*
3. *Street Markets*

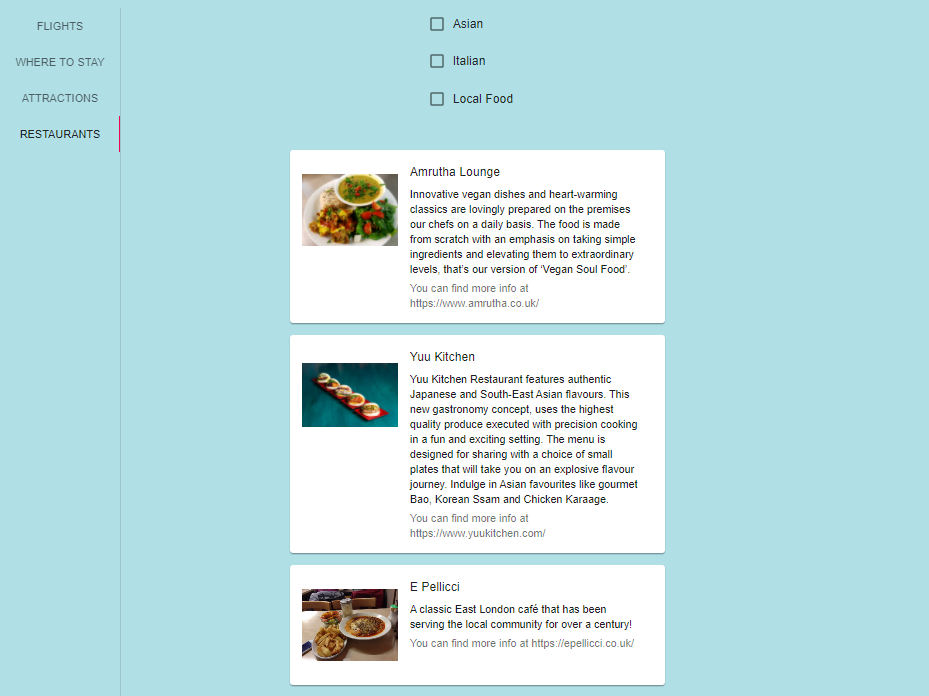
הוא עושה זאת באמצעות *component* מסוג *FormControl*, היוצר בתוכו את התגיות הרלוונטיות אותן ניתן לסמן ב-וי או לבטל את הסימון, וכך בעצם להציג את האטרקציות שהמשתמש רוצה. הדבר מתבצע בעזרת הקטגוריות המוקצות לכל אובייקט של אטרקציה במערך האטרקציות. הסינון מתבצע כך:  
כל קטגוריה מיוצגת בעזרת *component* מסוג *FormControlLabel*, וכאשר היא מסומנת על ידי המשתמש, מופעלת הפונקציה *handleCheck* שמוסיפה את הקטגוריה הזו למערך ריק בשם *filteredTags* שנמצא ב-*state*. וכן אם המשתמש ביטל את הסימון לקטגוריה מסוימת, הפונקציה מוציאה אותה מהמערך שב-*state*. מערך האטרקציות שמועבר ל-*CardsGrid*, עובר תמיד דרך הפונקציה *filterArray*, אשר לוקחת את המערך ובעצם מסננת אותו לפי הקטגוריות שמופיעות במערך *filteredTags*. אם המערך ריק, היא פשוט מחזירה את המערך הרגיל, כיוון שאין קטגוריה שהמשתמש רוצה לסנן לפיה את האטרקציות. אם הוא לא ריק, הפונקציה משתמשת בפונקציה *filter* על המערך – היא בודקת כל אחת מהאטרקציות במערך, ואם לפחות אחת מהקטגוריות הרצויות מופיעה במערך הקטגוריות של האטרקציה, היא נכנסת למערך ובסופו של דבר מוצגת למשתמש.

בנוסף, ניתן לסמן ב'לייק' אטרקציות המועדפות על המשתמש.  
  


1. ***Restaurants* –** מכיל *component* מסוג *ItemsGrid*, המקבל בעזרת *props* מערך של מסעדות הנשלף בעזרת השרת (כאשר כל מסעדה היא אובייקט המכיל שם, תמונה ואינפורמציה), ולפיו הוא מוסיף פריטים המייצגים את המסעדות הללו – בעזרת *component* מסוג *Item*.

בנוסף, ובאותו האופן המתבצע ב-*component* מסוג *Attractions*, הוא מאפשר לסנן את המסעדות לפי קטגוריות שונות:

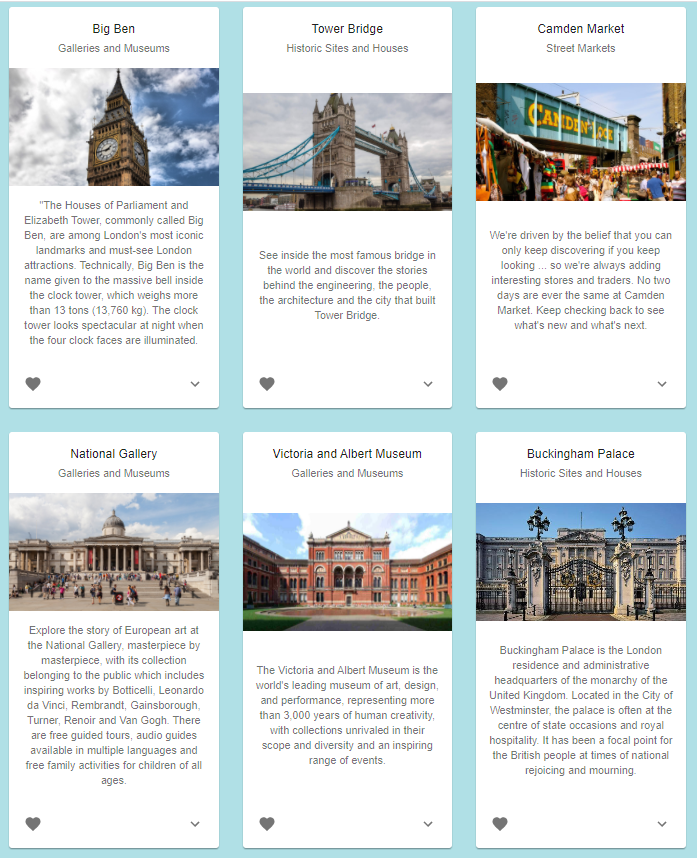
1. *Asian*
2. *Italian*
3. *Local Food*



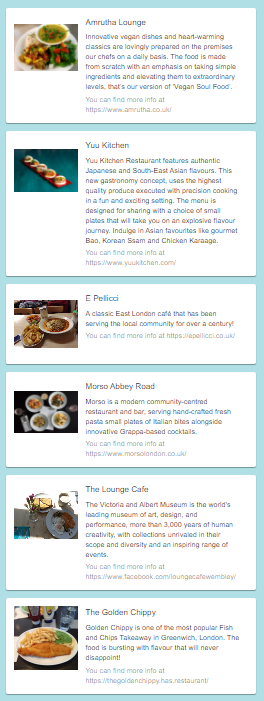
1. ***Facts* –** מקבל מערך של עובדות מהשרת ומציג אותן למשתמש.

****

1. ***CardsGrid* –** לוקח את המערך של האובייקטים שקיבל (במקרה הזה את מערך האטרקציות, יוצר *container* ובתוכו יוצר עבור כל איבר במערך *component* מסוג *Card*, כאשר כל *component* כזה מייצג כרטיסייה של אטרקציה.



1. ***Card* –** מקבל אובייקט המייצג אטרקציה, לוקח את המידע שהוא מכיל ומייצג אותו בצורת כרטיסייה – שם, כותרת משנה, תמונה, אינפורמציה ראשונית ואינפורמציה נוספת: אפשר ללחוץ על החץ הקטן מצד ימין למטה ולקבל את הקישור לאתר הבית. בנוסף, ניתן ללחוץ על אייקון הלב ולסמן כך אטרקציה/מסעדה מועדפת.  
     
     
2. ***ItemsGrid*** **–** לוקח את המערך של האובייקטים שקיבל (במקרה הזה את מערך המסעדות), יוצר *list* ובתוכה יוצר עבור כל איבר במערך *component* מסוג *Item*, כאשר כל *component* כזה מייצג פריט מסעדה.



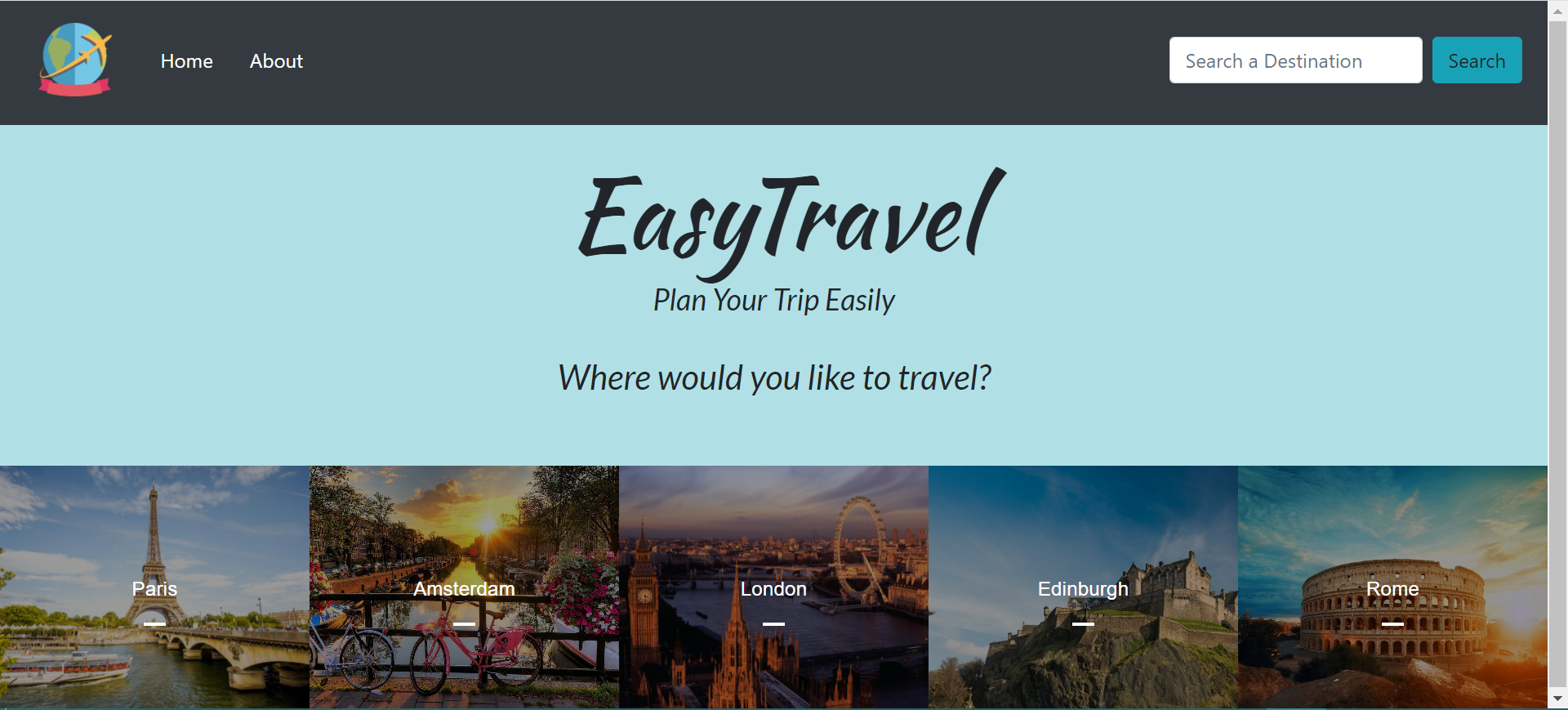
1. ***Item*** **–** מקבל אובייקט המייצג מסעדה, לוקח את המידע שהוא מכיל ומייצג אותו בצורת מלבן – שם, תמונה, אינפורמציה ראשונית ואינפורמציה נוספת.



ממשק המשתמש

*מסך ראשי – דף הבית*

ניתן לבחור מבין 5 ערים שונות את היעד הרצוי. בנוסף, בשורת הניווט ניתן לבצע חיפוש של העיר הרצויה (וכן ניתן לעשות זאת בכל אחד מהמסכים). בלחיצה על *Home* בשורת הניווט מכל אחד מהמסכים ניתן להגיע חזרה לדף הבית, ובלחיצה על *About* בשורת הניווט ניתן לקבל מידע על האתר עצמו.

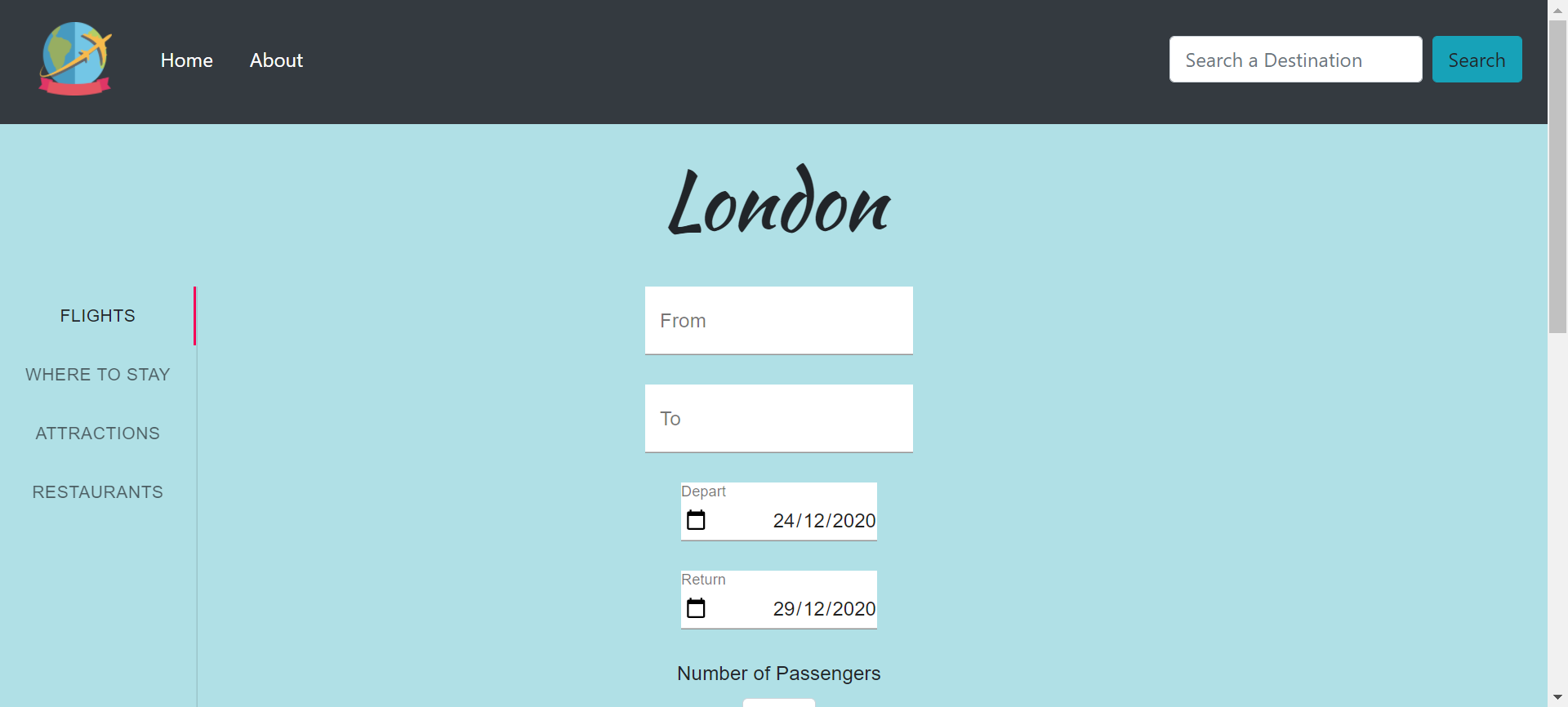


*מסך About*



*מסך של עיר ספציפית*

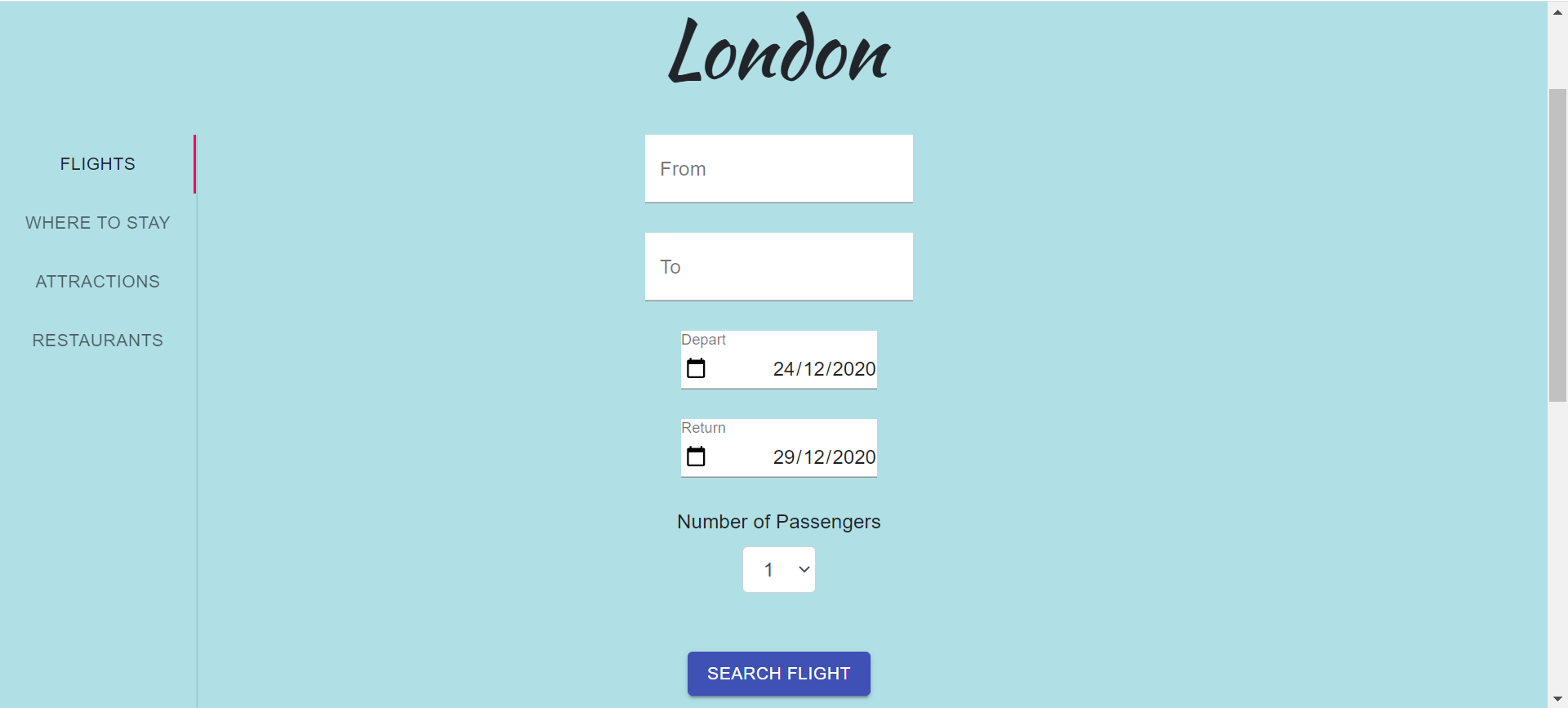
לכל עיר יש מסך משלה ובו לשוניות עבור טיסות לאותה העיר, מקומות לינה, ואטרקציות ומסעדות מומלצות.



הלשוניות (*tabs*):

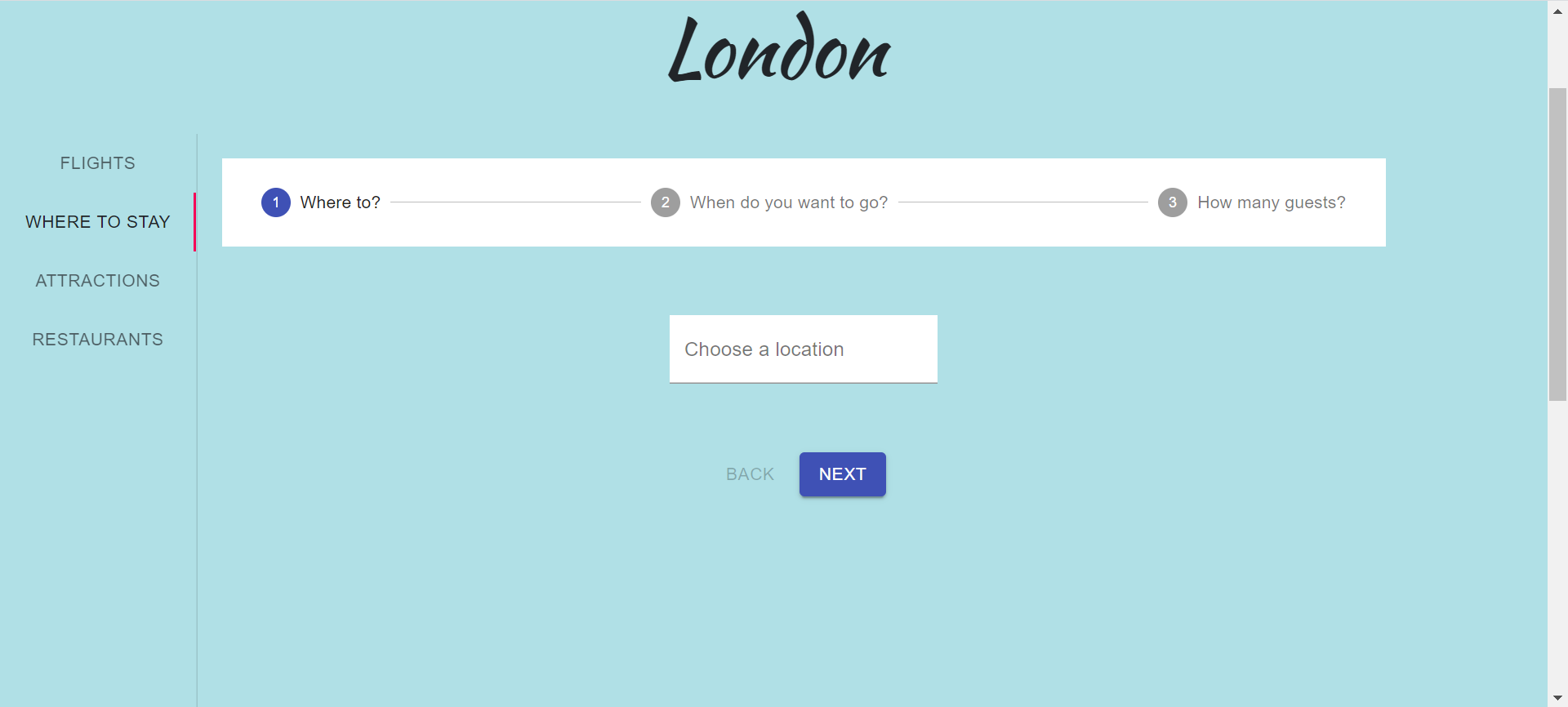
*טיסות*

כאן אפשר למלא את הנתונים הרצויים בשדות: מהיכן טסים, תאריך יציאה, תאריך חזרה ומספר הנוסעים.

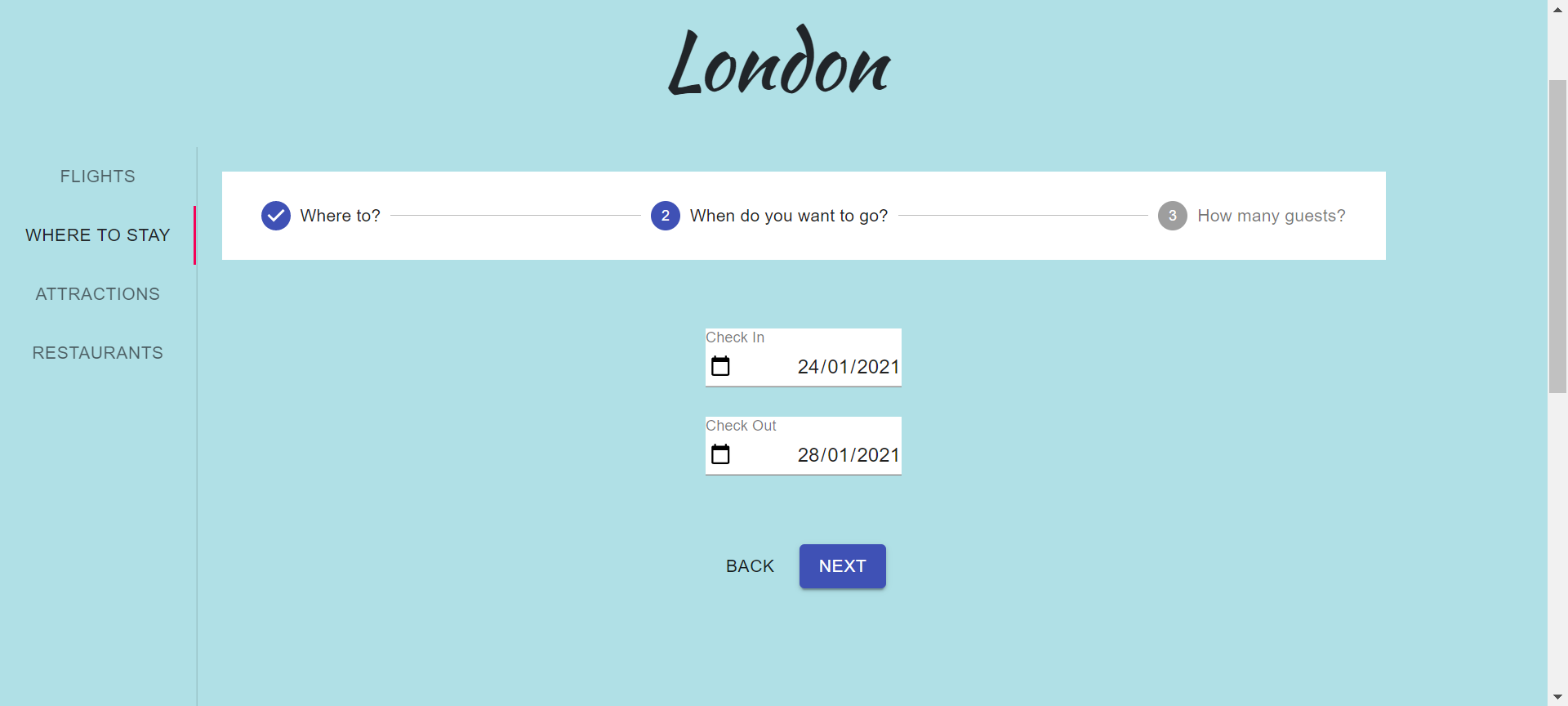


*מקומות לינה*

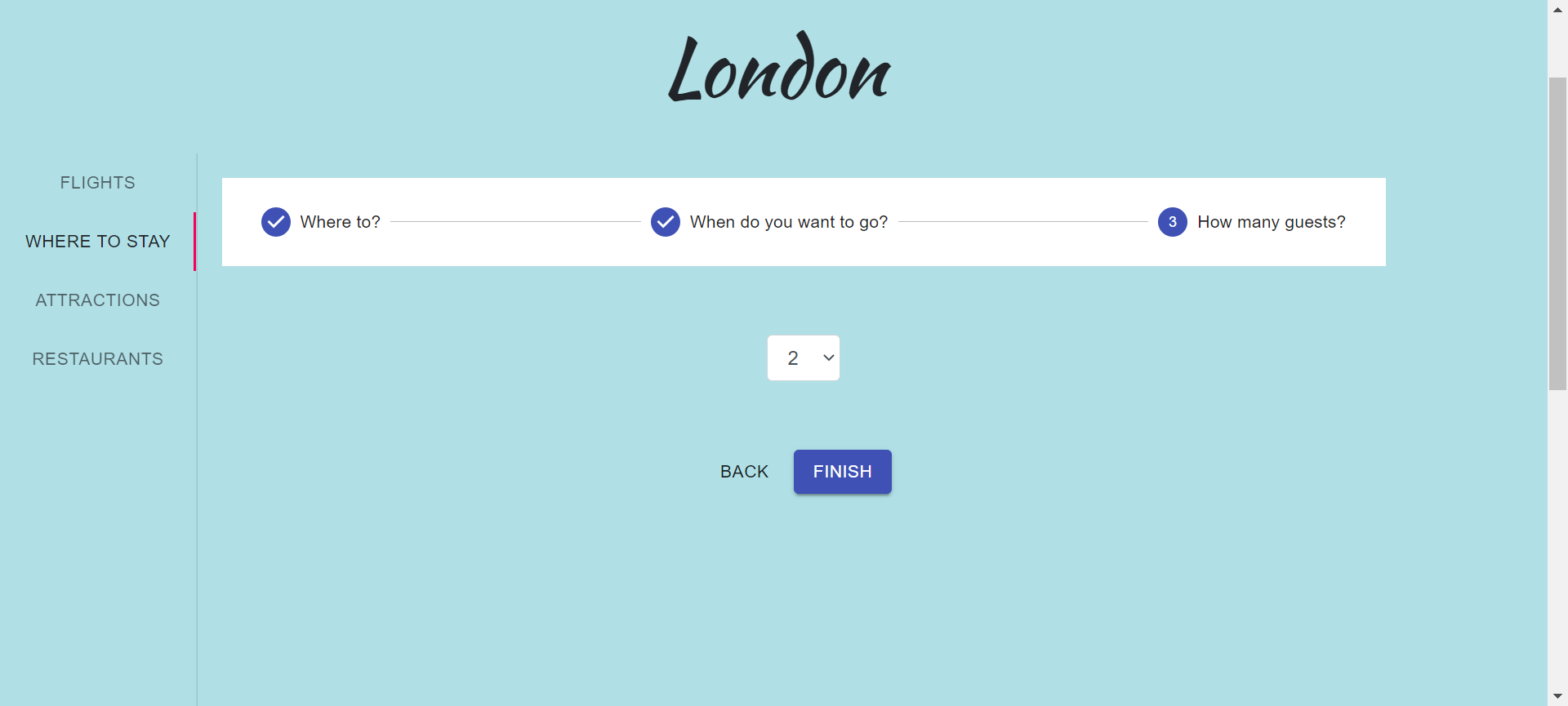
כאן אפשר למלא את הנתונים הרצויים בשדות, בכמה שלבים: בשלב הראשון בוחרים את המיקום בו רוצים ללון:



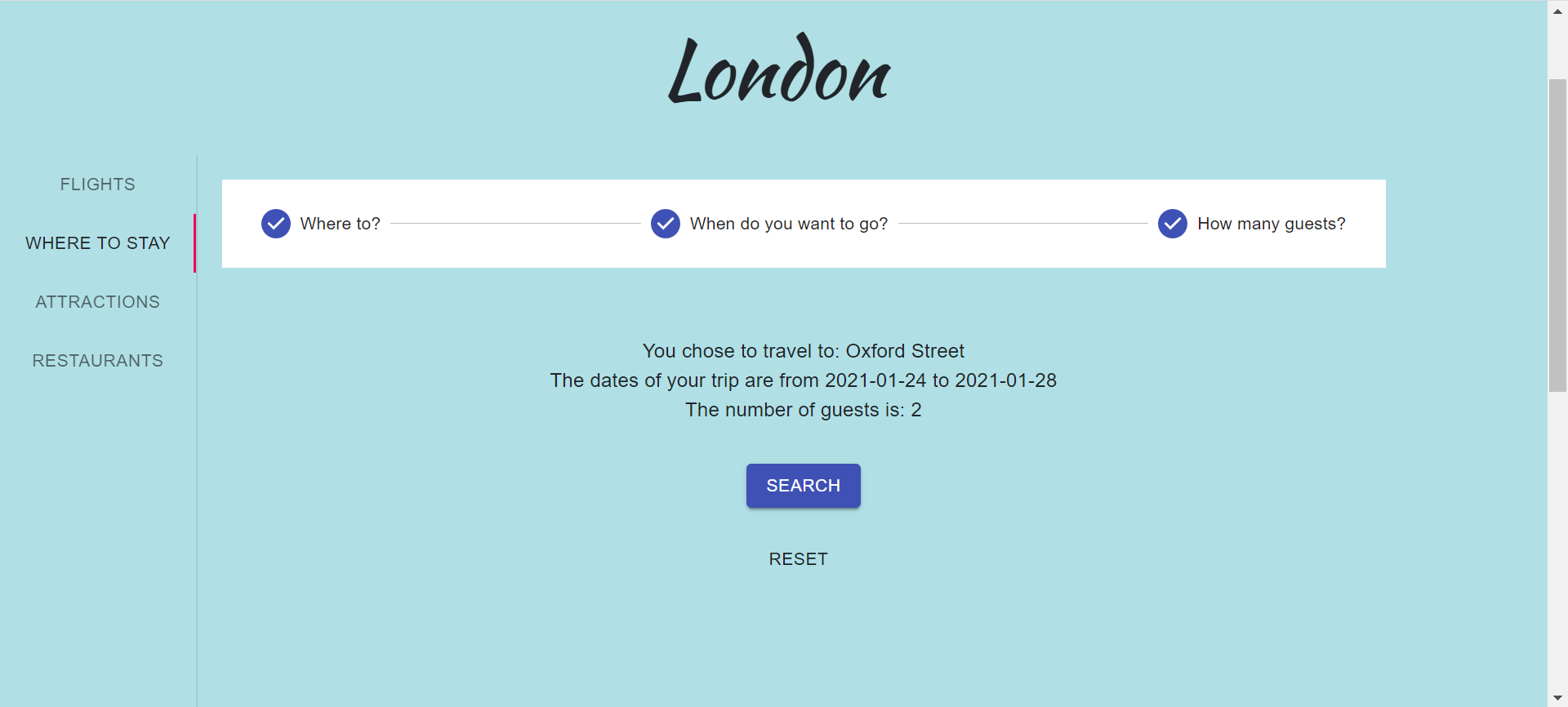
בשלב השני בוחרים את תאריך הצ'ק אין ואת תאריך הצ'ק אאוט:



בשלב האחרון בוחרים את מספר האורחים:

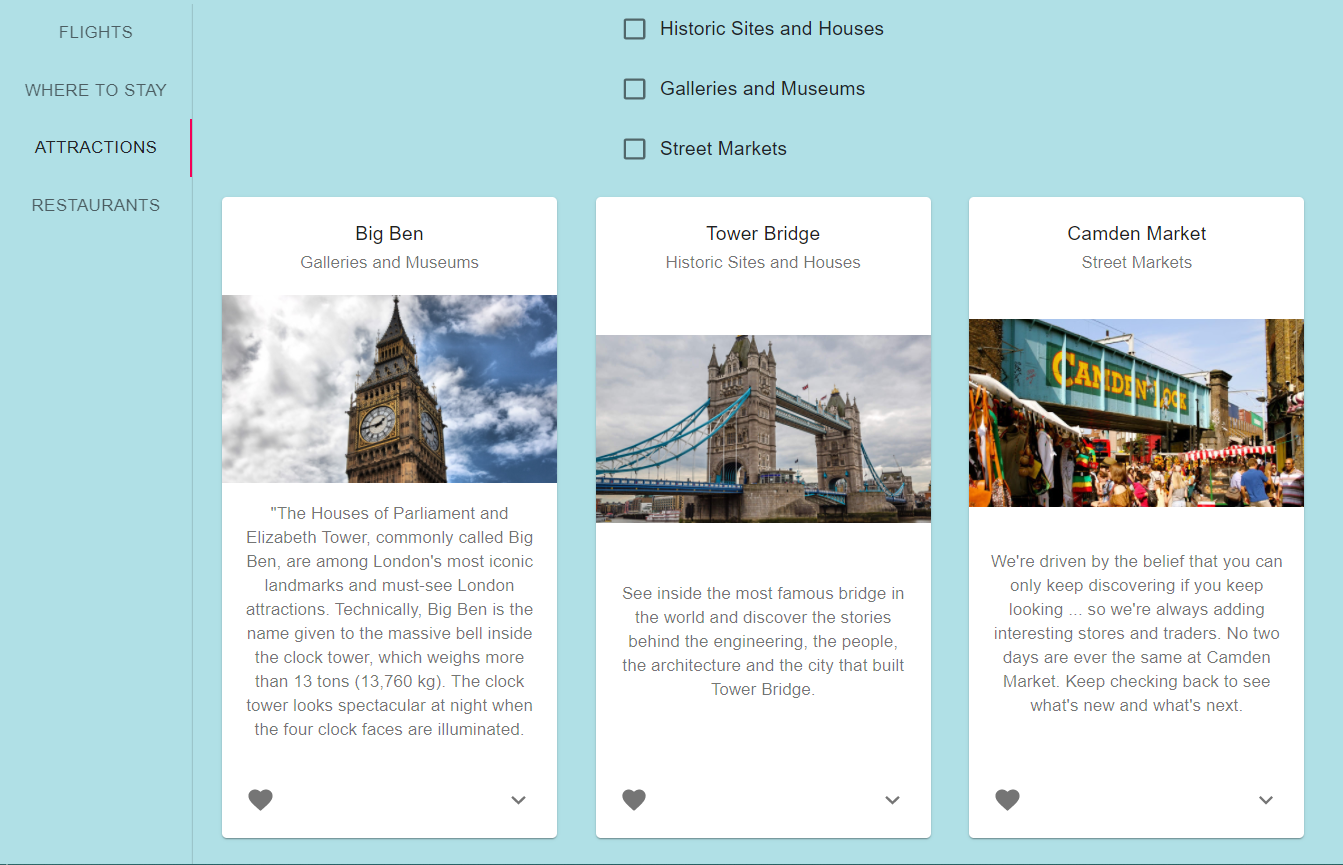


כעת ניתן לראות את הבחירות של המשתמש בכל השלבים, ואם ירצה לשנות את הפרטים יוכל לאתחל את הבחירה ולשנות את הפרטים:



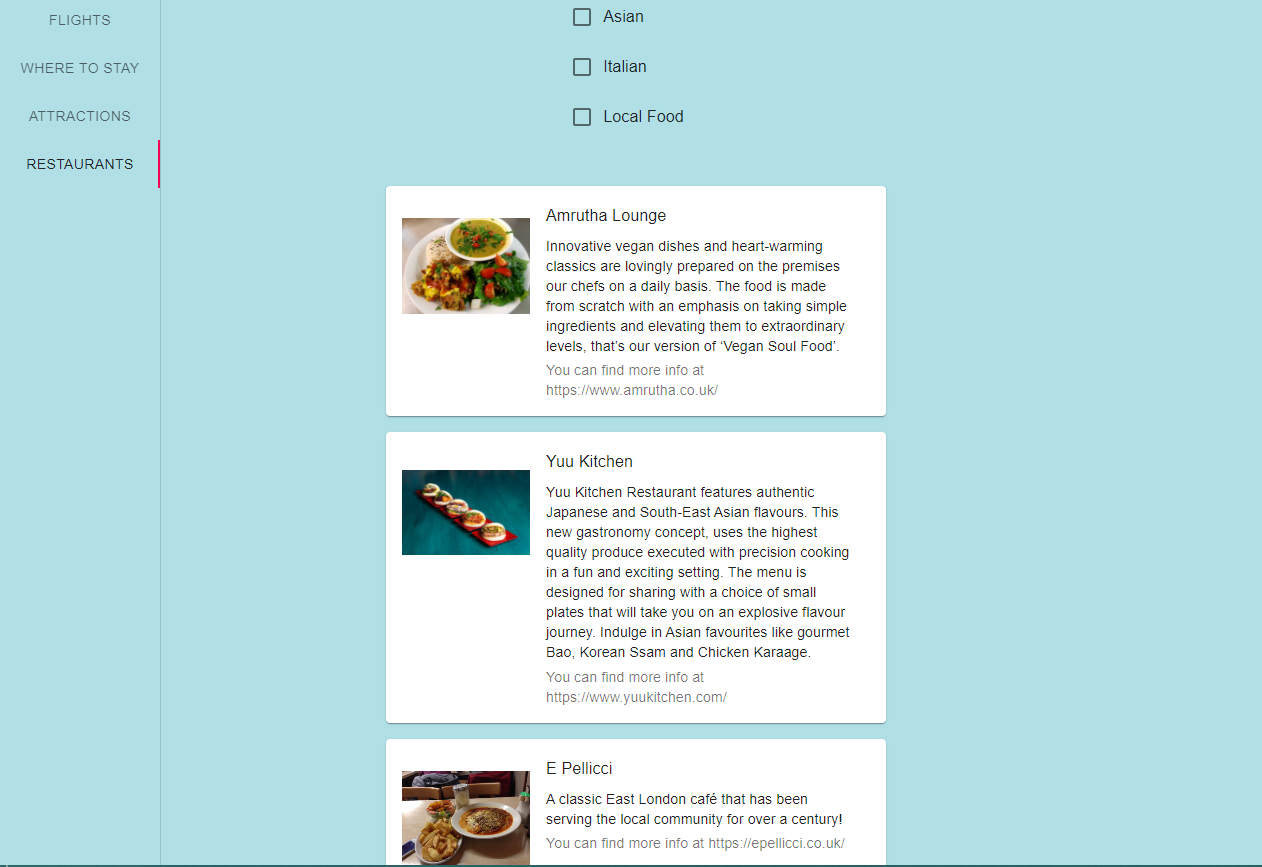
*אטרקציות*

בלשונית זו נוכל למצוא את כל האטרקציות המומלצות באותה עיר ולסנן אותן לפי קטגוריה מועדפת. עבור כל אטרקציה ניתן לראות לאיזו קטגוריה היא משתייכת, תמונה ותיאור קצר. בנוסף, ניתן לקבל קישור לאתר המכיל פרטים נוספים על האטרקציה ולסמן את האטרקציות המועדפות.



*מסעדות*

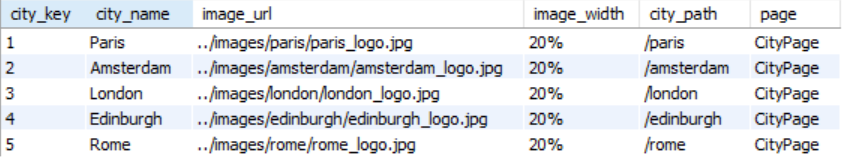
בלשונית זו נוכל למצוא את כל המסעדות המומלצות באותה עיר ולסנן אותן לפי קטגוריה מועדפת. עבור כל מסעדה ניתן לראות תמונה מייצגת ותיאור קצר, וכן קישור למידע נוסף באתר המסעדה.



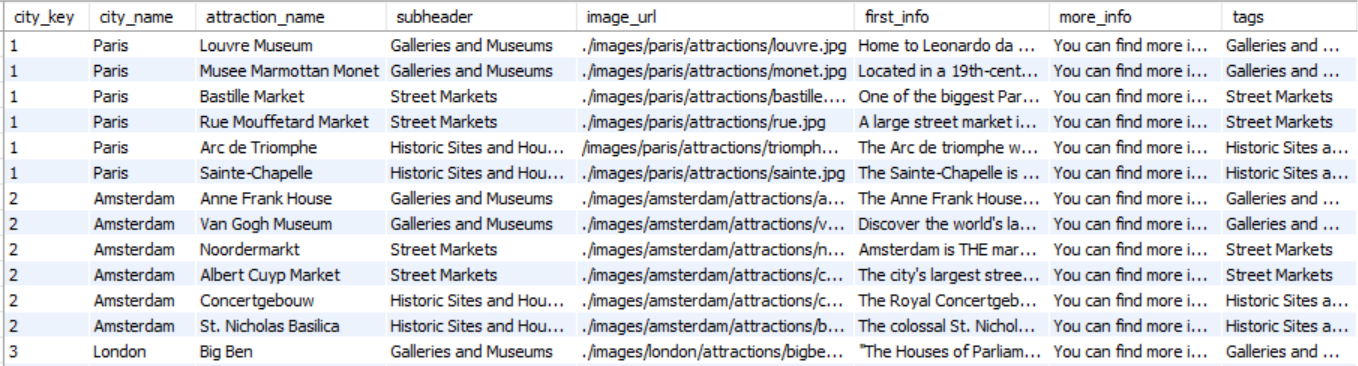
ניהול הנתונים

הנתונים כוללים את:

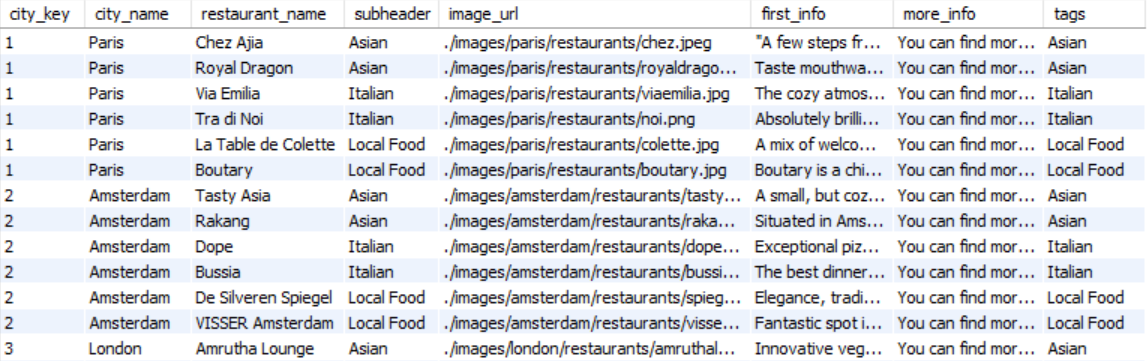
1. שמות הערים השונות הקיימות באתר ואינפורמציה נוספת הקשורה אליהן:



1. האטרקציות המומלצות של כלל הערים באתר ואינפורמציה נוספת הקשורה אליהן (בתמונה ניתן לראות חלק מהטבלה):



1. המסעדות המומלצות של כלל הערים באתר ואינפורמציה נוספת הקשורה אליהן (בתמונה ניתן לראות חלק מהטבלה):



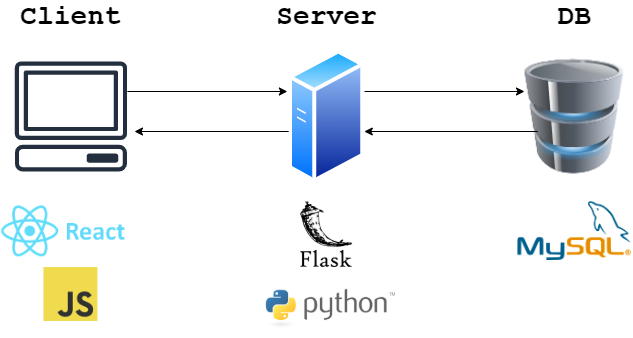
1. העובדות על כל עיר (בתמונה ניתן לראות חלק מהטבלה):



הנתונים מאוחסנים ב-*DB*, במודל *SQL*, בעזרת מערכת *MySQL*.

שילוב מערכות אחרות

השרת הוקם באמצעות סביבת הפיתוח *flask* בשפת *Python*. צד הלקוח שולח לו בקשות לקבלת הנתונים הרלוונטיים, הוא מתקשר עם ה-*DB* וניגש אליו, שולף את הנתונים הרלוונטיים ושולח אותם חזרה לצד הלקוח על מנת שיוצגו למשתמש באתר.



**MVP**

התכולות שמימשתי במסגרת הזמן הנתון

* הקמת צד שרת לאתר.
* קריאה מ-*DB* בשפת *SQL* ושליפה של המידע הנחוץ לאתר.
* המסך הראשי בו המשתמש יכול לבחור את היעד הרצוי.
* מסך *About*.
* מסך המייצג יעד מסוים.
* לשוניתשל חיפוש טיסות (מימוש צד הלקוח בלבד).
* לשונית של חיפוש מקומות לינה (מימוש צד הלקוח בלבד).
* לשוניתשל האטרקציות המומלצות בכל יעד, כאשר כל אטרקציה מוצגת בצורת כרטיסייה עם הפרטים שלה: כותרת, סוג האטרקציה, תמונה מייצגת, אינפורמציה ראשונית ואינפורמציה נוספת.
* אפשרות לסנן את האטרקציות לפי קטגוריות מסוימות.
* אפשרות לסמן ב'לייק' אטרקציות המועדפות על המשתמש.
* לשוניתשל המסעדות המומלצות בכל יעד, כאשר כל מסעדה מוצגת בצורת מלבן עם הפרטים שלה: כותרת, תמונה מייצגת, אינפורמציה ראשונית ואינפורמציה נוספת.
* אפשרות לסנן את המסעדות לפי קטגוריות מסוימות.
* אפשרות חיפוש יעד באתר.

למימוש עתידי

* אפשרות רישום של משתמשים בעזרת שם משתמש/כתובת מייל וסיסמא.
* הוספת יעדים נוספים ואינפורמציה נוספת על כל יעד, כמו מידע על תחבורה, כסף מקומי וכדו'.
* חיפוש מיקום הלינה בעזרת *API* של *Google Maps*.
* הוספת *API* המאפשר את שליחת הנתונים שהמשתמש הזין בחיפוש הטיסות ומקומות הלינה ישירות לאתרי החיפוש כגון *Booking* ו-*SkyScanner*.
* אפשרות לשמור ברשימה פרטית של המשתמש אטרקציות ומסעדות המועדפות עליו.
* אפשרות דירוג של אטרקציות ומסעדות על ידי משתמשים והצגת הדירוג המשוקלל של כלל המשתמשים.
* הוספת *autocomplete* לחיפוש.