פרויקט קורס WEB

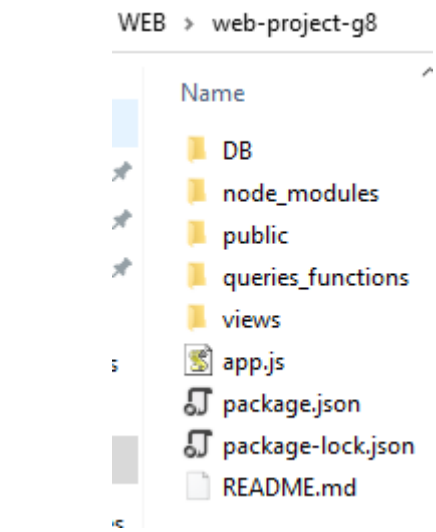
חלק ב'

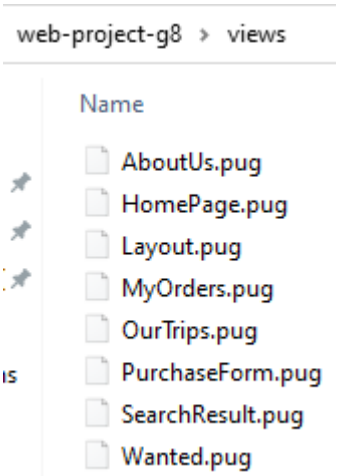
קבוצה 8

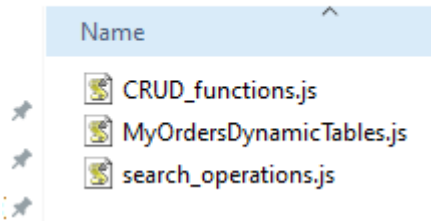
מגישים: 315330985, 209108885, 314628868, 211636451

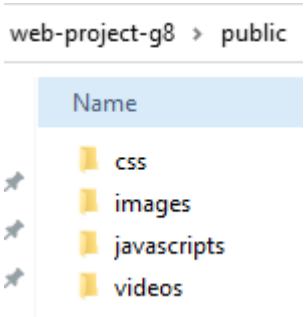
תוכן

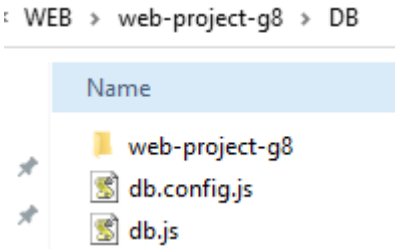
מבנה תיקיית הפרויקט:











חיבור לבסיס הנתונים "web-project-g8" :

קבצי החיבור לבסיס הנתונים כמו כן קובץ בסיס הנתונים נמצאים בתוך תיקיית "DB". על מנת לייבא את בסיס הנתונים (טבלאות + רשומות) יש לייבא את כל התיקייה web-project-g8/DB/web-project-g8.

*בנספחים מוצג הסקריפט של בניית הטבלאות והכנסת הרשומות.

שאילתות:

1. הכנסת לקוח חדש (createNewClient)- הרשמה של משתמש חדש במערכת, על מנת לבצע רכישה של כרטיסים על המשתמש להיות קיים ומחובר למערכת. מתבצע בטופס registration.

```
sql.query("INSERT INTO clients SET ?", newClient, (err, mysqlres) => {
    if (err) {
```

2. התחברות לאתר (LogIn) – השאילתה בודקת אם אכן שם המשתמש שהוכנס הינו בעל הסיסמא הנכונה. קריאה מבסיס הנתונים.

```
const LogIn = function(request, response) {
    var loginClient = {
        "username": request.body.username,
        "password": request.body.psw,

    };
    if (loginClient.username && loginClient.password) {
        sql.query('SELECT * FROM clients WHERE user_name=? AND password = ?', [loginClient.username, loginClient.password], f
```

3. ביטול רכישת כרטיס טיסה (cancelFlight)- כל רכישה נשמרת בטבלת passengers (email,ID).
השאילתה מוחקת מהטבלה את כרטיסה הטיסה הרלוונטי.

```
sql.query("delete from passengers where email=? and ID=?", [req.session.userid, shuttleID], (err, mysqlres) => {
    if (err) {
        console.log("error: ", err);
```

4. חיפוש טיסות לפי קטגוריות חיפוש (searchmenu)- בעמוד הבית בביצוע חיפוש, בהתאם לקלטי המשתמש (האם המשתמש הכניס רק מיקום נוכחי ויעד או גם תאריכי טיסה הלוך וחזור) תוציא השאילתה את התוצאות הרלוונטיות מבסיס הנתונים.

```
sql.query('SELECT * FROM shuttles WHERE current_location = ? AND destination= ?', [Fl
    if (error) {
        console.log("error: ", err);
        response.status(400).send({ message: "error in finding shuttles: " + err });
        return;
```

```
else {
  sql.query('SELECT * FROM shuttles WHERE current_location = ? AND destination = ? AND departure_date=?', [FlightsSearch.curre
    console.log("search with depurture date");
    if (error) {
        console.log("error: ", err);
```

5. עדכון היסטורית החיפוש- בכל פעם שלקוח מחובר מבצע חיפוש מתעדכנת טבלת "search_history" על היעד אותו הלקוח חיפוש וכמות הפעמים שחיפש אותו.

```
if (req.session.userid) {
    console.log("IN THIS FUNCTION");
    console.log(req.session.userid);
    const email = req.session.userid;
    sql.query('SELECT * FROM search_history WHERE email = ?', email, function (error, resu
        const searchObj = {
            email,
            destination: req.query.DestinationL
        }
        if (results.length === 0) {
            searchObj.search_number = 1;
            sql.query("INSERT INTO search_history SET ?", searchObj, (err, mysqlres) => {
                if (err) {
                    console.log("error: ", err);
                    return;
                }
                console.log("created searchObj");
                return;
            });
        } else {
            searchObj.search_number = results[0].search_number + 1;
            sql.query("UPDATE search_history SET ? WHERE email = ?", [searchObj, email], (
                if (err) {
                    console.log("error: ", err);
                    return;
                }
                console.log("updated searchObj");
```

6. בחירת כרטיסים לרכישה(Purchaseform)- בעת בחירת כרטיסים מתוצאות החיפוש על ידי checkbox, השאילתה מוצאיה את פרטי הטיסות מבסיס הנתונים, מציגה אותם ומחשבת מחיר כולל.

```
if (DepurtureShuttel_ID && ReturnShuttel_ID) {

    var DepurtureShuttel = [];
    var ReturnShuttel = [];

    sql.query('SELECT * FROM shuttles WHERE ID=?', DepurtureShuttel_ID, f
        if (err) {
            console.log(err);
        }
        if (results.length > 0) {
            for (var i = 0; i < results.length; i++) {
                var cDepurtureShuttel = {
                    'ID': results[i].ID,
                    'from': results[i].destination,
                    'to': results[i].current_location,
                    'depurtuedate': results[i].departure_date,
                    'capacity': results[i].capacity, // todo culc functio
                    'price': results[i].ticket_price,
                }
                DepurtureShuttel.push(cDepurtureShuttel);


            }

        }
        if (ReturnShuttel_ID) {
            sql.query('SELECT * FROM shuttles WHERE ID=?', ReturnShuttel_
```

7. ביצוע קנייה (MakePurchase)- מבצעים בדיקה אם ללקוח כבר קיימים פרטי האשראי בבסיס הנתונים, אם לא, מכניסים אשראי חדש. אחר כך מעדכנים את טבלת passengers בברטיסי הטיסה והלקוח אשר רכש אותם.

```
query('SELECT * FROM credit_cards WHERE credit_number = ? AND email =?', [addCredit.credit_number, addCredit.email],

    if (results.length>0) {
        console.log("credit card already in use with client");



    } else {
        sql.query("INSERT INTO credit_cards SET ?", addCredit, (err, mysqlres) => {
            if (err) {
```

```
var passenger1 = { 'email': req.session.userid, 'ID': DepurtureShuttel_ID };
var passenger2 = { 'email': req.session.userid, 'ID': ReturnShuttel_ID };
if (DepurtureShuttel_ID) {
    sql.query("INSERT INTO passengers SET ? ", passenger1, (err, mysqlres) =>
        if (err) {
```

*שאילתה נוספת **מסונכרנת** מתבצעת עבור מעבר על כלל הרכישות של אותו משתמש מחובר על מנת להציג את כלל הטיסות שלו בקריאה לעמוד "my orders". מוצגת במס'7 .

8. ריענון "הטיסות שלי" (TablesOnload)- בכל כניסה לעמוד "my orders" נקראת שאילתה במוציאה מבסיס הנתונים את כלל הטיסה אליה רשום המשתמש המחובר. שאילתה זו נקראת גם בסוף תהליך רכישה חדשה ובתהליך מחיקה של טיסה אליה רשום.

```
const TablesOnload = async function (email) {
    let orders = [];
    let Shuttel;
    if (email) {
        const shuttles = await query('SELECT * FROM shuttles as s join passengers as p on s.ID=p.ID WHERE p.email = ? group by s.ID order by 4', email);
        if (shuttles.length > 0) {
            for (var j = 0; j < shuttles.length; j++) {
                Shuttel = {
                    'ID': shuttles[j].ID,
                    'from': shuttles[j].destination,
                    'to': shuttles[j].current_location,
                    'depurtuedate': new Date(shuttles[j].departure_date).toLocaleDateString(),
                    'price': shuttles[j].ticket_price,
                }
                orders.push(Shuttel);
            }
            return { Shuttel, orders };;
        }
    }
    return { Shuttel, orders };
};

module.exports = { TablesOnload };
```

9. getTrips- השאילתה עוברת על טבלת הטיסות ומחזירה אותן ממוינות לפי יעדים ולפי תאריכי המראה.

```
const getTrips = async function () {
    const locations_trips = {};
    const locations = await query('SELECT * FROM locations');
    for (let i = 0; i < locations.length; i++) {
        location = locations[i].location;
        const trips = await query('SELECT * FROM shuttles WHERE destination=?', location);
        trips.forEach(trip => {
            trip.departure_date_formatted = new Date(trip.departure_date).toLocaleDateString();
        });
        locations_trips[location] = trips;
    }
    return { locations, locations_trips };
```

10. findReturnTrips- עבור בחירת טיסה מיעד ספציפי בעמוד "our trips" , השאילתה תחפש טיסה חזור מאותו יעד ומתאריך חזרה החל מתאריך הטיסה הלוך.

```
const findReturnTrips = async function (location, date) {
    const trips = await query('SELECT * FROM shuttles WHERE destination=? AND departure_date > ?', [location, date]);
    return trips.map((trip) => {
        return {
            ID: trip.ID,
            from: trip.current_location,
            to: trip.destination,
            departureDate: new Date(trip.departure_date).toLocaleDateString(),
            capacity: trip.capacity,
            price: trip.ticket_price
```

11. הכנסת בקשה חדשה לעבודה (createNewWanted) – הכנסת רשומה חדשה לבסיס הנתונים בטבלת wanted.

```
    }
    const newWanted = {
        "first_name": req.body.fname,
        "last_name": req.body.lname,
        "phone_number": req.body.phone,
        "email": req.body.email,
        "submitted_job": req.body.job
    };
    sql.query("INSERT INTO wanted SET ?", newWanted, (err, mysqlres) =>
        if (err) {
```

12. יצירת קשר(createNewContact)- הכנסת רשומה חדשה לבסיס הנתונים בקשה ליצירת קשר

```
    }
    const newContact = {
        "email": req.body.email,
        "message": req.body.subject
    };

    sql.query("INSERT INTO contact_us SET ?", newContact, (err, mysqlres)
        if (err) {
```

מימוש טפסים:

1. **Log In**

את טופס ההתחברות קורא ל 'action='http://localhost:3000/auth', method='post
המבצע בדיקה כי אכן המשתמש והסיסמה קיימים ונכונים ובנוסף מעדכן על ידי חבילת express-session מי המשתמש שהתחבר לאתר.
" session.userid= LoggedInUser[0].email;"
הטופס ממומש בעמוד ה "layout" ובסיום המימוש חוזר לעמוד ממנו נקרא.

2. **Registration**

הרשמת משתמש חדש מתבצע ב 'action='http://localhost:3000/newClient', method='post
הטופס ממומש בעמוד ה "layout" ובסיום המימוש חוזר לעמוד ממנו נקרא.

3. **Search result**

*הטופס במקור הוצג בעמוד הבית, בחלק ג' הומר לעמוד pug נפרד
(action='http://localhost:3000/purchaseForm', method='post')
הטופס מקבל את תוצאות החיפוש של הלקוח ומציג את הטיסות בטבלאות, כאשר הלקוח בוחר על ידי check box את הטיסה אותה רוכש הטופס שולח אותו המשך תהליך הרכישה כאשר מוצא את פרטי הטיסה ומחשב את סך עלות הכרטיסים.

4. **purchaseForm**

*הטופס במקור היה רק popup והומר לעמוד pug.
(action='http://localhost:3000/myorders', method='post')
הטופס מבצע הכנסת פרטי אשראי במידה ועוד לא קיימים והרשמת הלקוח לטיסות אותן רכש. בסוף תהליך הרכישה שולח הטופס את המשתמש לעמוד "my orders" אשר מציג את כלל הכרטיסים של הלקוח כולל החדשים שהוספו.

5. **Contact us**

(action='http://localhost:3000/contactResult', method='post')
יצירת קשר מכניסה רשומה חדשה לטבלת "contact_us" עם מייל הפונה ותיבת טקסט חופשי.

6. **Our trips**

(method='post', action='http://localhost:3000/searchResult')
בעמוד "our trips" מציג הטופס עבור כל יעד טבלה ממוינת לפי תאריכי המראה.
כאשר לקוח בוחר טיסה הלוך הטופס שולח אותו לעמוד "search results" ומאפשר לו לבחור את הטיסה חזרה מאותו יעד נבחר.

7. **Recommended suttles**

(action='http://localhost:3000/searchResult', method='post')
בעמוד הבית מוצגות 3 טיסות מומלצות על ידי האתר (עם פונקציונליות מורכבת המוצגת למטה). גם כאן, בעת בחירה של טיסה מסויימת, ישלח הטופס לעמוד "search results" המאפשר ללקוח לבחור את הטיסה חזרה מאותו יעד נבחר.

8. **Wanted**

(action='http://localhost:3000/wantedResult', method='post')
בעמוד "wanted" הטופס מאפשר הכנסה של בקשה לעבודה עבור תפקיד נבחר ספציפי באותו המוד.
הכנסת רשומה חדשה לטבלת "wanted".

מימוש פונקציונאליות עיבוד מידע:

getTopShuttles()- בכל פעולת חיפוש שלקוח מחובר עושה, נשמרת היסטורית החיפוש בבסיס הנתונים בתוך טבלת "search_history" (שאילתה 5). הפונקציה **מחזירה את שלושת היעדים שהלקוח חיפש הכי הרבה** ומציגה אותם בעמוד הבית בטופס "future shuttles: recommended for you".

במצבים בהם אין לקוח מחובר לאתר או אין היסטורית חיפוש ללקוח או ללקוח פחות משלושה יעדים בהיסטורית החיפוש: תציג הפונקציה כברירת מחדל הצעות לכרטיסי טיסה הממוינים לפי תאריך ההמראה הקרוב ביותר (שהחברה שלנו רוצה למכור מהר כאינטרס עסקי)

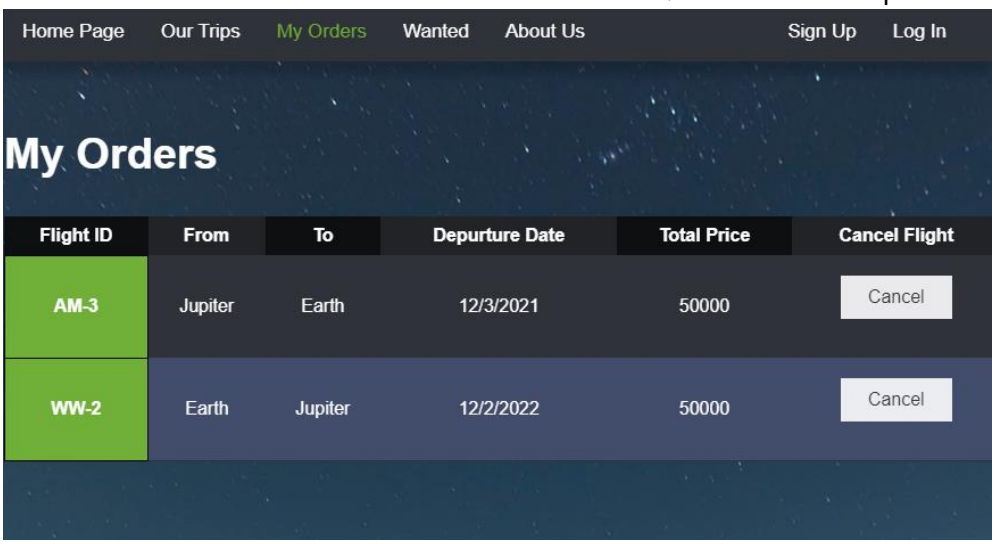
```
const getTopShutteles = async function (email) {
    let trips = [];
    if (email) {

        const topSearchs = await query('SELECT * FROM search_history WHERE email = ? ORDER BY search_number LIMIT 1', email);
        trips = await query(`SELECT * FROM shuttles ORDER BY departure_date LIMIT 3`);
        for (var i = 0; i < topSearchs.length; i++) {
            const trip = await query('SELECT * FROM shuttles WHERE destination = ? ORDER BY departure_date LIMIT 1', topSearchs[i].destination);
            if (trip.length > 0) {
                trips[i] = trip[0];
            }
        }
        return trips;
    }
    trips =  await query('SELECT * FROM shuttles ORDER BY departure_date LIMIT 3');
    trips.forEach(trip => {
        trip.departure_date_formatted = new Date(trip.departure_date).toLocaleDateString();
    });
    return trips;
```

תוכן דינמי - PUG:

עם העברת עמודי הhtml שלנו לpug, יצרנו תוכן דינמי:

1. "My Orders" – העמוד מציג את כל רכישות המשתמש בטבלה המתעדכנת מבסיס הנתונים בכל ריענון הדף ובכל פעולת רכישה או ביטול.
במידה ולקוח לא מחובר לא יוצגו נתונים.



2. "Recommended flights – על ידי מימוש פונקציונלי של היעדים שחיפש הלקוח הכי הרבה, עמוד הבית מציג את שלוש הטיסות המומלצות לאותו לקוח או טיסות היוצאות בזמן הכי קרוב. הצגת יעדים אלה דינמיים ומתעדכנים בכל ריענון של אתר הבית בהתאם ללקוח המחובר.

נספחים:

טבלאות בסיס הנתונים:

clients(email,user_name, password, first_name, last_name, birthdate)

locations(location)

shuttles(ID, current_location(locations), destination(locations), departure_date, ticket_price, capacity)

passengers(email,ID)

credit_cards(credit_namber, exp_month, exp_year, cvv,email(clients))

search_history(email, search_number, destination)

contact_us(email,message)

wanted(first_name,last_name,phone_number,email,submitted_job)

סקריפט בסיס נתונים:

```
drop table clients;

drop table credit_cards;

drop table locations;

drop table shuttles;

drop table passengers;

drop table search_history;

drop table contact_us;

drop table wanted;


create table clients (

email varchar(255) primary key,

user_name varchar(255)  not null,

password varchar(255)   not null,

first_name varchar(255)   not null,

last_name varchar(255)  not null,

birthdate date  not null

) engine=innoDB default charset=utf8;


insert into clients(email,user_name, password, first_name, last_name, birthdate) values
("baraklavi2009@gmail.com","barak","123456barak", “barak”, “lavi”, “1995-05-22”),

("baraklavi200@gmail.com","barak1","123456barak", “barak”, “lavi”, “1995-05-22”),

("baraklavi20@gmail.com","barak2","123456barak", “barak”, “lavi”, “1995-05-22”),

("baraklavi2@gmail.com","barak3","123456barak", “barak”, “lavi”, “1995-05-22”),

("baraklavi@gmail.com","barak4","123456barak", “barak”, “lavi”, “1995-05-22”);
```

```
create table credit_cards (
credit_number varchar(255) primary key,
exp_month  varchar(255) not null,
exp_year varchar(255) not null,
cvv varchar(255) not null,
email varchar(255) not null
) engine=innoDB default charset=utf8;

insert into credit_cards(credit_number, exp_month, exp_year, cvv,email) values
(“123456789”,”07”,”2025”,”555”,"baraklavi2009@gmail.com"),
(“12345678910”,”07”,”2025”,”555”,"baraklavi20@gmail.com"),
(“1234567890”,”07”,”2025”,”555”,"baraklavi200@gmail.com"),
(“1234567896”,”07”,”2025”,”555”,"baraklavi2009@gmail.com"),
(“1234567895”,”07”,”2025”,”555”,"baraklavi@gmail.com"),
(“1234567894”,”07”,”2025”,”555”,"baraklavi200@gmail.com"),
(“1234567”,”07”,”2025”,”555”,"baraklavi2@gmail.com");
create table locations (
location varchar(30) primary key
) engine=innoDB default charset=utf8;
 insert into locations values (“Earth”),(“Mars”),(“Jupiter”),(“Venus”),(“ISS”);

create table shuttles(
ID varchar(255) primary key,
current_location varchar(30) not null,
destination varchar(30)  not null,
departure_date date null,
ticket_price int null,
capacity int null
) engine=innoDB default charset=utf8;
```

```
insert into shuttles(ID, current_location, destination, departure_date, ticket_price, capacity) values
(“AA-1”, “Earth”, “Mars”, “2021-11-01”, 20000, 200),
 (“AA-2”, “Earth”, “Mars”, “2021-11-02”, 20000, 200),
 (“AA-3”, “Earth”, “Mars”, “2021-11-03”, 20000, 200),
 (“AM-1”, “Earth”, “Venus”, “2021-12-01”, 30000, 200),
 (“AM-2”, “Earth”, “Jupiter”, “2021-12-02”, 50000, 200),
 (“AM-3”, “Earth”, “Jupiter”, “2021-12-03”, 50000, 200),
 (“AS-1”, “Earth”, “ISS”, “2021-11-01”, 15000, 500),
 (“AS-2”, “Earth”, “ISS”, “2021-11-02”, 15000, 500),
 (“AS-3”, “Earth”, “ISS”, “2021-11-03”, 15000, 500),
 (“AV-1”, “Earth”, “Venus”, “2021-11-01”, 20000, 200),
 (“AV-2”, “Earth”, “Venus”, “2021-11-02”, 20000, 200),
 (“AV-3”, “Earth”, “Venus”, “2021-11-03”, 20000, 200),
 (“AA-4”, “Earth”, “Mars”, “2021-11-04”, 20000, 200),
 (“AB-16”, “Mars”, “Earth”, “2023-11-01”, 20000, 200),
 (“BC-77”,  “Mars”, “Earth”, “2023-11-03”, 20000, 200),
 (“FB-5”,  “Mars”, “Earth”, “2023-11-05”, 20000, 200),
(“WW-1”, “Venus”, “Earth”, “2022-12-01”, 30000, 200),
 (“WW-2”,  “Jupiter”,“Earth”, “2022-12-02”, 50000, 200),
 (“WW-3”,“Jupiter”, “Earth”, “2023-12-03”, 50000, 200),
 (“RF-1”, “ISS”, “Earth”, “2021-12-01”, 15000, 500),
 (“ARFS-2”, “ISS”,“Earth”, “2021-12-02”, 15000, 500),
 (“RF-3”, “ISS”,  “Earth”,“2021-11-30”, 15000, 500),
 (“DE-1”,  “Venus”,“Earth”, “2023-11-01”, 20000, 200),
 (“SS-2”, “Venus”, “Earth”, “2024-11-02”, 20000, 200),
 (“VB-3”, “Venus”, “Earth”, “2024-11-03”, 20000, 200),
 (“VB-4”,“Mars”, “Earth”, “2023-11-04”, 20000, 200)
;
```

```
create table passengers (
email varchar(255) not null,
ID varchar(255) not null
) engine=innoDB default charset=utf8;
insert into passengers (email,ID) values ("baraklavi2009@gmail.com",”AS-1”);

create table search_history (
search_number int primary key,
email varchar(255) not null,
destination varchar(30) not null
) engine=innoDB default charset=utf8;

insert into search_history (search_number ,email,destination) values
(1,”baraklavi2009@gmail.com”,”ISS”);

create table contact_us (
email varchar(255) primary key,
message varchar(255) not null
) engine=innoDB default charset=utf8;

create table wanted(
first_name varchar(255) not null,
last_name varchar(255) not null,
phone_number varchar(255) not null,
email varchar(255) primary key,
submitted_job varchar(255) not null
) engine=innoDB default charset=utf8;
```