

# *ΕΡΓΑΣΙΑ ΜΑΘΗΜΑΤΟΣ*

---

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Πληροφορικής



Μάθημα: «ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΣΤΟ ΔΙΑΔΙΚΤΥΟ (8ο εξ.)»

Π18101 – ΑΝΑΣΤΑΣΙΑ ΙΩΑΝΝΑ ΜΕΞΑ

Π18078 – ΑΘΑΝΑΣΙΑ ΚΟΜΜΑΤΙΔΟΥ

Π18123 – ΒΑΣΙΛΙΚΗ ΠΑΣΙΑ



## Εκφώνηση της άσκησης

Ανάπτυξη Διαδικτυακού Πληροφοριακού Συστήματος

Η εργασία είναι υποχρεωτική και έχει βαρύτητα 70% επί της συνολικής βαθμολογίας.

Στην εργασία η κάθε ομάδα, η οποία θα αποτελείται από δύο άτομα το πολύ, θα πρέπει να κατασκευάσει μια web εφαρμογή με JSP/Servlets (εναλλακτικά με C# .NET). Ειδικότερα, εκτός των βασικών λειτουργιών που περιγράφονται παρακάτω, θα πρέπει να αναπτύξετε μια αλγοριθμική διαδικασία που θα βρίσκεται στον πυρήνα της εφαρμογής.

### Βασικές λειτουργίες:

1. Δημιουργία βάσης δεδομένων.
2. Δημιουργία χρηστών.
3. Διαχείριση χρηστών και κανόνες ελέγχου πρόσβασης. Διαφορετικές κατηγορίες χρηστών θα πρέπει να έχουν και διαφορετικά δικαιώματα πρόσβασης σε συγκεκριμένα αρχεία και δεδομένα.
4. Παρουσίαση και επεξεργασία των δεδομένων που βρίσκονται αποθηκευμένα στη βάση.

### Αλγοριθμική διαδικασία

Η εφαρμογή που θα αναπτύξετε θα πρέπει να προσφέρει στους χρήστες της χρήσιμα συμπεράσματα και πληροφορίες. Ένα παράδειγμα αποτελεί μια εφαρμογή που αποφασίζει αν θα πρέπει κάποιος πελάτης να λάβει μια χρηματοπιστωτική υπηρεσία. Σε αυτήν την περίπτωση, η απόφαση θα ληφθεί βάσει δεδομένων που εισάγονται από τον χρήστη ή που βρίσκονται στη βάση αλλά και δεδομένων που ανακτώνται μέσω web services από άλλους οργανισμούς. Συνεπώς, στο πλαίσιο της εφαρμογής θα πρέπει να αναπτύξετε την προαναφερθείσα αλγοριθμική διαδικασία που οδηγεί στη συγκεκριμένη απόφαση. Περισσότερα και αναλυτικότερα παραδείγματα παρουσιάζονται στο μάθημα.

### Παραδοτέα

Στα παραδοτέα συμπεριλαμβάνονται τα ακόλουθα:

- Κώδικας της εφαρμογής και η βάση δεδομένων σε ηλεκτρονική μορφή.
- Ένα τεύχος στο οποίο θα περιγράφονται οι απαιτήσεις και η λειτουργικότητα της εφαρμογής. Επιπροσθέτως, σε παραρτήματα θα πρέπει να δοθεί ο τρόπος υλοποίησης (κώδικας), καθώς και ένα user manual με τη χρήση ενδεικτικών screenshots.
- Η παράδοση της εργασίας θα συνοδεύεται απαραίτητως από επίδειξη της λειτουργίας της, όπου θα αποδεικνύεται ότι η εφαρμογή λειτουργεί χωρίς προβλήματα.



## ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1	Απαιτήσεις και λειτουργικότητα της εφαρμογής .....	4
2	Βάση .....	4
3	Κλάσεις .....	5
3.1	Η κλάση Meals .....	5
3.2	Η κλάση User .....	6
3.3	Η κλάση Auxiliary .....	8
4	Λειτουργία του Spoonacular API.....	10
4.1	Παραδείγματα JSON requests στο API με την χρήση Postman.....	10
5	Χάρτης .....	14
5.1	Υποστηριζόμενες κουζίνες από το Spoonacular API.....	16
5.2	Επεξήγηση κώδικα της εφαρμογής .....	17
6	Αποθήκευση πλάνου και αγαπημένων συνταγών.....	19
6.1	Επεξήγηση κώδικα αποθήκευσης πλάνων .....	19
6.2	Επεξήγηση κώδικα αποθήκευσης αγαπημένων συνταγών .....	21
7	Εισαγωγή της ιστοσελίδας στον IIS και δημιουργία πιστοποιητικού.....	22
7.1	Προσθήκη της εφαρμογής στον IIS.....	22
7.2	Δημιουργία SSL certificate .....	23
8	Εγχειρίδιο χρήστη .....	25
8.1	Σύντομη παρουσίαση του προγράμματος .....	25
9	Βιβλιογραφικές Πηγές.....	34

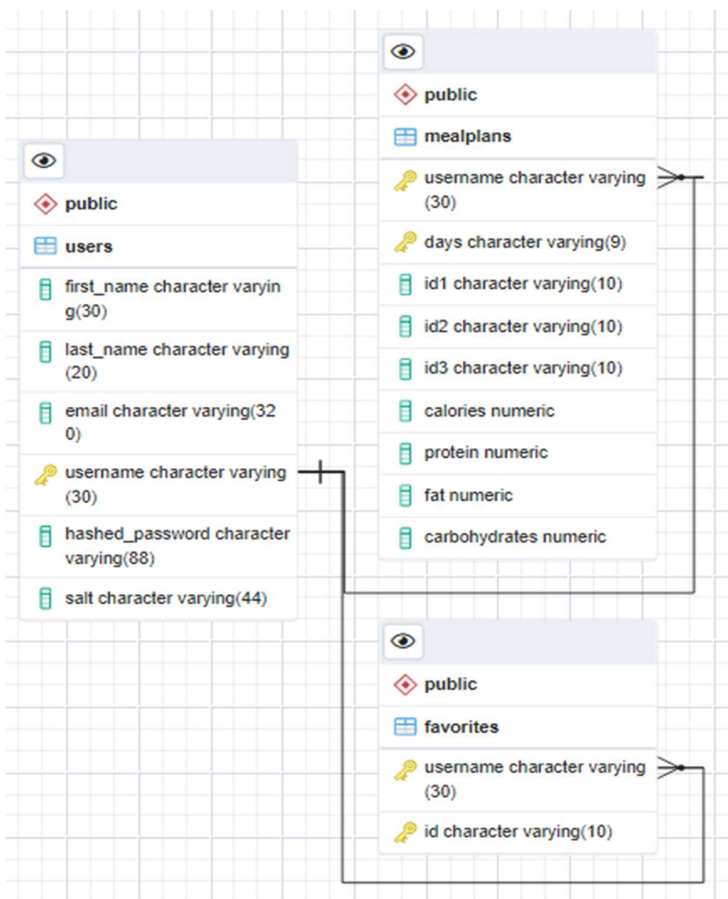
## 1 Απαιτήσεις και λειτουργικότητα της εφαρμογής

Στο πλαίσιο του μαθήματος αναπτύξαμε μια διαδικτυακή εφαρμογή με το όνομα «Foodies», στην οποία οι χρήστες της μπορούν να αναζητούν συνταγές βάσει κάποιων συγκεκριμένων φίλτρων που θα επιλέγουν (π.χ. υλικά, κουζίνα, κλπ.). Επίσης, υπάρχει η δυνατότητα να αποθηκεύουν αγαπημένες συνταγές και να δημιουργούν ημερήσιο ή/και εβδομαδιαίο πλάνο για μέτρηση θερμίδων και κόστους. Τέλος, έχουμε υλοποιήσει και wine recommendation, μια λειτουργία εύρεσης κατάλληλου κρασιού με βάση κάποιο φαγητό, ή αντίστοιχα εύρεσης προτεινόμενου φαγητού με βάση το κρασί της επιλογής τους.

Για την υλοποίηση της εφαρμογής χρησιμοποιήσαμε το περιβάλλον του Visual Studio σε γλώσσα C# χρησιμοποιώντας τεχνολογία ASP.NET. Η βάση δεδομένων που χρησιμοποιήθηκε είναι η PostgreSQL. Επιπλέον, έγινε χρήση του Spoonacular API (<https://spoonacular.com/food-api>) για την εύρεση των συνταγών/κρασιών.

## 2 Βάση

Η βάση μας ονομάζεται foodies και ο κώδικας για την δημιουργία των πινάκων βρίσκεται στο αρχείο `CREATE_CODE.txt`



Εικόνα 1: Σχήμα της βάσης

- Πίνακας users:

Σε αυτόν τον πίνακα αποθηκεύονται οι πληροφορίες των χρηστών μας. Συγκεκριμένα αποθηκεύουμε το όνομα (first\_name), επώνυμο (last\_name), email, username (πρωτεύον κλειδί) και τον κωδικό τους ο οποίος είναι σε κρυπτογραφημένη μορφή με την χρήση hash και salt. Ο αλγόριθμος που χρησιμοποιείται για την κρυπτογράφηση είναι ο SHA512.

- Πίνακας mealplans:

Σε αυτόν τον πίνακα αποθηκεύονται πάντα τα τελευταία εβδομαδιαία και ημερήσια πλάνα των χρηστών. Πρωτεύον κλειδί είναι ο συνδυασμός του username του χρήστη και της στήλης days, δηλαδή υπάρχουν μοναδικά ζεύγη χρήστη και ημέρας (π.χ. (testuser,monday), (testuser,today), κλπ.). Σημειώνεται ότι το username είναι foreign key που αφορά τον πίνακα των χρηστών.

- Πίνακας favorites:

Σε αυτόν τον πίνακα αποθηκεύονται οι αγαπημένες συνταγές των χρηστών. Πρωτεύον κλειδί είναι ο συνδυασμός του username του χρήστη και του id της συνταγής που έχει ορίσει ως αγαπημένη. Σημειώνεται ότι το username είναι foreign key που αφορά τον πίνακα των χρηστών.



Ακολουθούν screenshots με ενδεικτικά παραδείγματα από εγγραφές των πινάκων.

	first_name character varying (30)	last_name character varying (20)	email character varying (320)	username [PK] character varying (30)	hashed_password character varying (88)	salt character varying (44)
1	Anastasia	Mexa	anastasia.mexa@yahoo.gr	stacy	Kx+lb7LivsuVOObXHj4lgnNz...	ww1+Jt36lE3G74P/s9Y5Xp...
2	Andreas	Mexas	andreas_mexas@yahoo.gr	andreas	vVUc1wPmSA5hcVQzYShjE...	xBAhWrfuQxEcC10pQUifGh...
3	Marianna	Eremia	lumis37@yahoo.gr	lumis	K7JLSvplyg4czAFEY594tdyt...	C5QJNlizzf/U44yQRbB/AYM...

Εικόνα 2: Εγγραφές πίνακα users

	username [PK] character varying (30)	days [PK] character varying (9)	id1 character varying (10)	id2 character varying (10)	id3 character varying (10)	calories numeric	protein numeric	fat numeric	carbohydrates numeric
1	stacy	Today	446790	248724	623480	1499.92	46.82	72.48	170.7
2	stacy	Monday	653019	1029741	1098240	1499.93	68.02	66.65	156.81
3	stacy	Tuesday	624585	552477	390800	1499.94	62.65	79.42	134.13
4	stacy	Wednesday	880442	219069	1075210	1499.94	65.55	107.46	67.72
5	stacy	Thursday	590870	602744	10423	1500	43.31	84.21	148.42
6	stacy	Friday	558647	769292	370641	1499.93	84.13	80.57	105.85
7	stacy	Saturday	177541	279511	663051	1499.95	83.98	51.62	172.53
8	stacy	Sunday	166789	470122	1521771	1499.96	53.81	114.57	63.57

Εικόνα 3: Εγγραφές πίνακα mealplans

	username [PK] character varying (30)	id [PK] character varying (10)
1	stacy	654959
2	stacy	654928
3	stacy	654812
4	stacy	511728
5	stacy	654926
6	stacy	642539

Εικόνα 4: Εγγραφές πίνακα favorites

### 3 Κλάσεις

Για την υλοποίηση των λειτουργιών που περιγράφηκαν παραπάνω, έχουμε δημιουργήσει τρεις κλάσεις που αφορούν τους χρήστες (*User.cs*), τα προγράμματα των γευμάτων (*Meals.cs*) και μια έξτρα βοηθητική κλάση που περιέχει μερικές μεθόδους και το connection string για την σύνδεση με την βάση (*Auxiliary.cs*). Ακολουθεί η ανάλυση των κλάσεων.

#### 3.1 Η κλάση Meals

Η συγκεκριμένη κλάση περιέχει μόνο τα πεδία και τον constructor. Η κύρια χρήση της είναι η δημιουργία αντικειμένων της, τα χαρακτηριστικά των οποίων θα αποθηκεύονται στην βάση μας.

```
public class Meals
{
    //---Fields---
    public string username;
    public string days;
    public string id1;
    public string id2;
    public string id3;
    public decimal calories;
    public decimal protein;
    public decimal fat;
    public decimal carbohydrates;

    //---Constructors---
    public Meals(string username, string days, string id1, string id2, string id3, decimal calories, decimal protein, decimal fat, decimal carbohydrates)
    {
        this.username = username;
        this.days = days;
        this.id1 = id1;
        this.id2 = id2;
        this.id3 = id3;
        this.calories = calories;
        this.protein = protein;
        this.fat = fat;
        this.carbohydrates = carbohydrates;
    }
}
```

Εικόνα 5: Κώδικας κλάσης Meals

### 3.2 Η κλάση User

Η συγκεκριμένη κλάση περιέχει τα πεδία και τους constructors που αφορούν τους χρήστες, καθώς και δύο επιπλέον μεθόδους, την StoreUserInfoToDB() η οποία αποθηκεύει έναν χρήστη στην βάση και την AuthenticateCredentials() η οποία επαληθεύει τα στοιχεία σύνδεσης του χρήστη.

```
public class User
{
    //---Fields---
    public string first_name;
    public string last_name;
    public string email;
    public string username;
    public string hashed_password;
    public string salt;

    //---Constructors---
    public User(string first_name, string last_name, string email, string username, string hashed_password, string salt)
    {
        this.first_name = first_name;
        this.last_name = last_name;
        this.email = email;
        this.username = username;
        this.hashed_password = hashed_password;
        this.salt = salt;
    }

    public User(string username)
    {
        this.username = username;
    }
}
```

Εικόνα 6: Πεδία και constructors της κλάσης User



- StoreUserInfoToDB()

Η μέθοδος εκτελεί ένα query στην βάση για την εισαγωγή των στοιχείων ενός χρήστη. Αρχικά, πραγματοποιείται η σύνδεση με την βάση μέσω του connection string που έχει οριστεί στην Auxiliary.cs (θα σχολιαστεί παρακάτω) και στην συνέχεια γίνεται χρήση parameterized queries για την εισαγωγή της εγγραφής στην βάση. Με αυτόν τον τρόπο, γίνεται προμεταγλώττιση του κώδικα SQL διαχωρίζοντάς τον από τα δεδομένα. Έτσι επιτυγχάνουμε την αποφυγή SQL injection επιθέσεων. Τέλος, εκτελείται το query και ελέγχουμε αν όλα έχουν πάει καλά με την χρήση try catch block.

```
//---Methods---  
//-----  
  
//Stores user's info(firstname, lastname, email, username,hashedpassword,salt) in DB when he signs up.  
public string StoreUserInfoToDB()  
{  
    string query = "insert into users values (@firstname, @lastname, @email, @username, @hashedPassword, @salt)";  
    int rowsAffected = -1; //false value  
    try  
    {  
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);  
        connection.Open();  
        //define query's parameters  
        NpgsqlCommand command = new NpgsqlCommand(query, connection);  
        command.Parameters.AddWithValue("firstname", first_name);  
        command.Parameters.AddWithValue("lastname", last_name);  
        command.Parameters.AddWithValue("email", email);  
        command.Parameters.AddWithValue("username", username);  
        command.Parameters.AddWithValue("hashedPassword", hashed_password);  
        command.Parameters.AddWithValue("salt", salt);  
        rowsAffected = command.ExecuteNonQuery(); //run query  
        connection.Close();  
    }  
    catch  
    {  
        return "Sign up failed. Please report this error through the \"Contact us\" page.";  
    }  
    if (rowsAffected == -1)  
        return "Sign up failed. There is a problem with the database. Please report this error through the \"Contact us\" page.";  
    else  
        return "done";  
}
```

Εικόνα 7: Κώδικας της μεθόδου StoreUserInfoToDB()

- AuthenticateCredentials()

Η μέθοδος εκτελεί ένα query στην βάση για την ανάκτηση των στοιχείων σύνδεσης (username, password) ενός χρήστη. Αρχικά, πραγματοποιείται η σύνδεση με την βάση μέσω του connection string που έχει οριστεί στην Auxiliary.cs και στην συνέχεια διαβάζουμε το username, το hashed\_password και το salt. Αν τα στοιχεία που έχει εισάγει ο χρήστης μέσω της φόρμας ταυτίζονται με αυτά της βάσης, τότε πραγματοποιείται επιτυχώς η σύνδεσή του, διαφορετικά εμφανίζεται κατάλληλο μήνυμα λάθους στον χρήστη.



```
//Used for logging in. Check if given username exists and if it does, then creates the hashed password anew and compares it with the one at the DB.
//If username and password match, then he logs in.
public string AuthenticateCredentials(string rawPassword)
{
    string query = "select username, hashed_password, salt from users";
    List<string> userData = new List<string>(); //store user's username, hashed password and salt
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query
        while (dataReader.Read())
            userData.Add(dataReader[0].ToString() + "|" + dataReader[1].ToString() + "|" + dataReader[2].ToString());
        connection.Close();
    }
    catch (Exception e)
    {
        return e.Message;
    }
    foreach (string dataRow in userData)
    {
        string[] dataCols = dataRow.Split('|');
        if (dataCols[0].Equals(username))
            if (dataCols[1].Equals(Auxiliary.HashPassword(rawPassword, Convert.FromBase64String(dataCols[2]))))
                return null;
            else
                return "Incorrect password.";
    }
    return "Username does not exist.";
}
//=====
```

Εικόνα 8: Κώδικας της μεθόδου AuthenticateCredentials()

### 3.3 Η κλάση Auxiliary

Η συγκεκριμένη κλάση περιέχει το connection string για την βάση, καθώς και μερικές έξτρα βοηθητικές μεθόδους που χρησιμοποιούνται κατά την διαδικασία της σύνδεσης και της εγγραφής ενός χρήστη.

```
public class Auxiliary
{
    public static readonly string CONNECTION_STRING = "Host=127.0.0.1;Port=5432;User ID=postgres;Password=maxrouso2;Database=foodies;";
}
```

Εικόνα 9: Connection string

Πιο συγκεκριμένα οι μέθοδοι αυτές είναι:

- GenerateSalt()

Η μέθοδος δημιουργεί ένα salt μήκους 32 byte.

```
//---Hash Password---
//=====

//Generate a salt to hash the user's password.
public static byte[] GenerateSalt()
{
    byte[] salt = new byte[32]; //salt length is 32 (can be changed)
    using (var random = new RNGCryptoServiceProvider())
    {
        random.GetNonZeroBytes(salt); //salt
    }
    return salt;
}
```

Εικόνα 10: Κώδικας της μεθόδου GenerateSalt()



- HashPassword()

Η μέθοδος δημιουργεί το hashed password του χρήστη μαζί με το salt που παράχθηκε από την προηγούμενη μέθοδο.

```
//Hashes the user's password along with the salt.
public static string HashPassword(string rawPassword, byte[] salt)
{
    byte[] password = Encoding.UTF8.GetBytes(rawPassword); //convert to bytes
    byte[] passwordWithSalt = new byte[password.Length + salt.Length];
    for (int i = 0; i < password.Length; i++) //copy password bytes
    {
        passwordWithSalt[i] = password[i];
    }
    for (int i = 0; i < salt.Length; i++) //copy salt bytes
    {
        passwordWithSalt[i + password.Length] = salt[i];
    }
    using (SHA512 shaM = new SHA512Managed())
    {
        return Convert.ToBase64String(shaM.ComputeHash(passwordWithSalt)); //hashed password (with salt)
    }
}
//=====
```

Εικόνα 11: Κώδικας της μεθόδου HashPassword()

- CheckForDuplicates()

Η μέθοδος ελέγχει αν υπάρχουν ήδη τα στοιχεία ενός χρήστη στην βάση, ώστε να αποφευχθούν τυχόν διπλότυπα στοιχεία.

```
//----Sign Up----
//=====

//Username is pk, email is unique
public static string CheckForDuplicates(string email, string username)
{
    string query = "select email, username from users";
    List<string> emailsDB = new List<string>(); //store emails
    List<string> usernamesDB = new List<string>(); //store usernames
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(CONNECTION_STRING);
        connection.Open();
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query
        while (dataReader.Read())
        {
            emailsDB.Add(dataReader[0].ToString());
            usernamesDB.Add(dataReader[1].ToString());
        }
        connection.Close();
    }
    catch
    {
        return "An exeption was caught. Please report it.";
    }
    foreach (string emailDB in emailsDB) //check for duplicates (email)
    {
        if (emailDB.Equals(email))
            return "Email already exists";
    }
    foreach (string usernameDB in usernamesDB) //check for duplicates (username)
    {
        if (usernameDB.Equals(username))
            return "Username already exists";
    }
    return null;
}
//=====
```

Εικόνα 12: Κώδικας της μεθόδου CheckForDuplicates()

- IsValidUserInputLogin()

Η μέθοδος ελέγχει την εγκυρότητα των στοιχείων σύνδεσης του χρήστη.

```
//---Log In---  
//=====
```

```
//Validate user's input.  
public static string IsValidUserInputLogin(string username, string password)  
{  
    if (username.Trim().Length == 0)  
        return "Please fill all input fields";  
    if (password.Trim().Length < 6)  
        return "Password must be at least 6 characters long";  
    return null;  
}  
//=====
```

Εικόνα 13: Κώδικας της μεθόδου IsValidUserInputLogin()

## 4 Λειτουργία του Spoonacular API

Καθ' όλη την διάρκεια της εφαρμογής, ότι πληροφορία χρειαζόμαστε σχετικά με τις συνταγές/γεύματα/κρασιά, την αντλούμε μέσω JSON request από το Spoonacular API (<https://spoonacular.com/food-api>). Για να μπορέσουμε να έχουμε πρόσβαση στα δεδομένα του API, κληθήκαμε να δημιουργήσουμε έναν λογαριασμό για να αποκτήσουμε ένα API key, το οποίο εισάγουμε στο url κάθε request. Να σημειωθεί ότι χωρίς συνδρομή, έχουμε περιορισμένο αριθμό request που μπορούμε να κάνουμε ανά μέρα.

### 4.1 Παραδείγματα JSON requests στο API με την χρήση Postman

Για να γίνει πλήρως αντιληπτή η λειτουργία επικοινωνίας μας με το API, αποφασίσαμε να περιγράψουμε μερικά ενδεικτικά JSON request, αναλύοντας την δομή του url και της απάντησης που δεχόμαστε. Φυσικά για το πλήρες documentation του API μπορείτε να επισκεφθείτε τον σύνδεσμο <https://spoonacular.com/food-api/docs>.

#### Search Recipes:

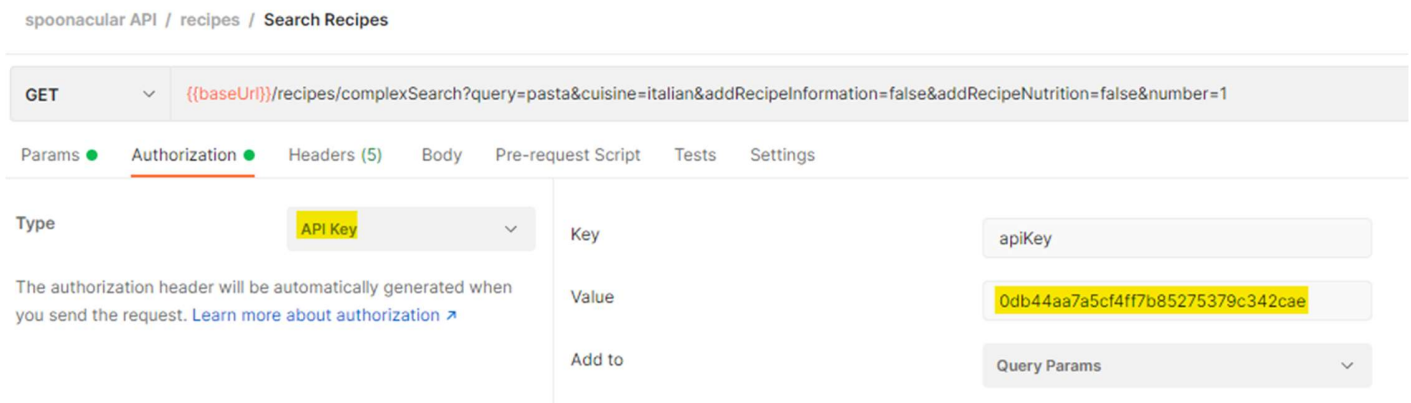
Για την αναζήτηση συνταγών στο API πρέπει η δομή του url να έχει την εξής μορφή:

<https://api.spoonacular.com/recipes/complexSearch>

Σημείωση: Στα επόμενα παραδείγματα το πορτοκαλί κομμάτι του url θα το συμβολίσουμε ως `{{baseUrl}}`, διότι το θεωρούμε ως βάση η οποία παραμένει η ίδια για όλα τα requests.

Βεβαίως, μπορούμε να ορίσουμε μερικές παραμέτρους για την εξατομίκευση των αποτελεσμάτων. Για παράδειγμα, μπορούμε να ορίσουμε βασικά υλικά των συνταγών με την χρήση της παραμέτρου “query” και τι είδους κουζίνα θέλουμε με την παράμετρο “cuisine”. Επίσης, αν το επιθυμούμε, μπορούμε να ορίσουμε τιμές στις παραμέτρους “addRecipeInformation” και “addRecipeNutrition”, αναλόγως αν θέλουμε να μας επιστραφούν επιπλέον πληροφορίες για τις συνταγές ή ακόμα και πληροφορίες που αφορούν τα θρεπτικά συστατικά των συνταγών. Τέλος, μπορούμε να ορίσουμε το επιθυμητό πλήθος των αποτελεσμάτων με την παράμετρο number.

Ακολουθούν δύο παραδείγματα JSON request από το περιβάλλον του Postman.



Εικόνα 14: Παράδειγμα 1 (Δομή url)

Σημείωση: Με τονισμένο κίτρινο χρώμα, απεικονίζεται το `apiKey` το οποίο χρησιμοποιούμε ως μηχανισμό αυθεντικοποίησης για να έχουμε πρόσβαση στα δεδομένα του API. Στα επόμενα παραδείγματα, η συγκεκριμένη πληροφορία θα παραλείπεται.

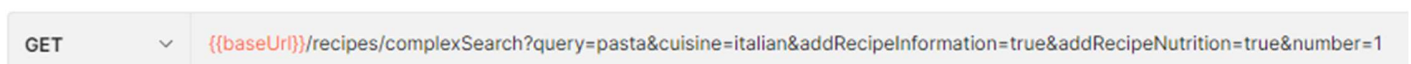
Όπως ορίζεται από τις τιμές των παραμέτρων του παραπάνω screenshot, αναζητούμε μια συνταγή ζυμαρικών ιταλικής κουζίνας, χωρίς έξτρα πληροφορίες για την συνταγή και για τα θρεπτικά συστατικά της. Το αποτέλεσμα του request φαίνεται στο επόμενο screenshot.

```
1  {
2    "results": [
3      {
4        "id": 654959,
5        "title": "Pasta With Tuna",
6        "image": "https://spoonacular.com/recipeImages/654959-312x231.jpg",
7        "imageType": "jpg"
8      }
9    ],
10   "offset": 0,
11   "number": 1,
12   "totalResults": 127
13 }
```

Εικόνα 15: Παράδειγμα 1 (Αποτέλεσμα request)

Μας έχουν επιστραφεί ο τίτλος της συνταγής, το id της και μία εικόνα της τα οποία περιέχονται στον πίνακα results. Ακόμα, μας ενημερώνει ότι πήραμε ένα από τα 127 συνολικά αποτελέσματα (συνταγές), που ικανοποιούν τα κριτήρια που θέσαμε.

Θα ξανακάνουμε το ίδιο request με την διαφορά ότι τώρα θέλουμε να λάβουμε και πληροφορίες για την συνταγή και τα θρεπτικά συστατικά της.



Εικόνα 16: Παράδειγμα 2 (Δομή url)

Το αποτέλεσμα του request φαίνεται στο επόμενο screenshot.

```
1  [
2  "results": [
3    {
4      "vegetarian": false,
5      "vegan": false,
6      "glutenFree": false,
7      "dairyFree": false,
8      "veryHealthy": true,
9      "cheap": false,
10     "veryPopular": false,
11     "sustainable": false,
12     "lowFodmap": false,
13     "weightWatcherSmartPoints": 11,
14     "gaps": "no",
15     "preparationMinutes": -1,
16     "cookingMinutes": -1,
17     "aggregateLikes": 2,
18     "healthScore": 89,
19     "pricePerServing": 168.12,
20     "id": 654959,
21     "title": "Pasta With Tuna",
22     "readyInMinutes": 45,
23     "servings": 4,
24     "sourceUrl": "http://www.foodista.com/recipe/K6QWSKQM/pasta-with-tuna",
25     "openLicense": -1,
26     "image": "https://spoonacular.com/recipeImages/654959-312x231.jpg",
27     "imageType": "jpg",
28     "nutrition": {
29       "nutrients": [
30         {
31           "name": "Calories",
32           "amount": 422.67,
33           "unit": "kcal",
34
35           "percentOfDailyNeeds": 21.13
36         },
37         {
38           "name": "Fat",
39           "amount": 10.32,
40           "unit": "g",
41           "percentOfDailyNeeds": 15.87
42         },
43         {
44           "name": "Saturated Fat",
45           "amount": 2.35,
46           "unit": "g",
47           "percentOfDailyNeeds": 14.69
48         },
49         {
50           "name": "Carbohydrates",
51           "amount": 57.66,
52           "unit": "g",
53           "percentOfDailyNeeds": 19.22
54         },
55         {
56           "name": "Net Carbohydrates",
57           "amount": 52.44,
58           "unit": "g",
59           "percentOfDailyNeeds": 19.07
60         },
61         ...
62       ]
63     }
64   ]
65 }
```

Εικόνα 17: Παράδειγμα 2 (Αποτέλεσμα request)

Παρατηρούμε πόσες περισσότερες πληροφορίες έχουμε λάβει, έχουν τονιστεί μερικές από της «σημαντικότερες» που χρησιμοποιούμε και εμείς στην εφαρμογή μας.

### Generate Meal Plan:

Για την δημιουργία πλάνων στο API πρέπει η δομή του url να έχει την εξής μορφή:

`{{baseUrl}}/mealplanner/generate`

Βεβαίως, μπορούμε να ορίσουμε μερικές παραμέτρους για την εξατομίκευση των αποτελεσμάτων. Για παράδειγμα, μπορούμε να ορίσουμε αν θέλουμε την δημιουργία εβδομαδιαίου ή ημερήσιου πλάνου, δίνοντας τιμή στην παράμετρο “timeFrame”. Επίσης, μπορούμε να θέσουμε το θερμιδικό στόχο που θέλουμε να πετύχουμε για κάθε μέρα του πλάνου με χρήση της παραμέτρου “targetCalories” και τέλος, μπορούμε να ορίσουμε μια συγκεκριμένη διαίτα που θέλουμε να ακολουθήσουμε (όπως vegetarian, vegan, pescetarian κλπ.) με την μεταβλητή “diet”.

Ακολουθούν δύο παραδείγματα JSON request από το περιβάλλον του Postman.

```
GET  {{baseUrl}}/mealplanner/generate?timeFrame=day&targetCalories=2000&diet=vegetarian
```

Εικόνα 18: Παράδειγμα ημερήσιου πλάνου (Δομή url)

Όπως ορίζεται από τις τιμές των παραμέτρων του παραπάνω screenshot, θέλουμε να παραχθεί ένα ημερήσιο πλάνο με στόχο τις 2.000 θερμίδες για μια χορτοφαγική διατροφή. Το αποτέλεσμα του request φαίνεται στο επόμενο screenshot.

```
2  "meals": [  
3    {  
4      "id": 833977,  
5      "imageType": "jpg",  
6      "title": "Honey-and-Tea Jammer Cookies",  
7      "readyInMinutes": 45,  
8      "servings": 30,  
9      "sourceUrl": "http://www.saveur.com/dorie-greenspan-jammer-cookies?dom=rss-default&src=syn"  
10    },  
11    {  
12      "id": 1152483,  
13      "imageType": "jpg",  
14      "title": "Chipotle-Lime Mixed Nuts",  
15      "readyInMinutes": 25,  
16      "servings": 4,  
17      "sourceUrl": "https://www.simplyrecipes.com/recipes/chipotle_lime_mixed_nuts/"  
18    },  
19    {  
20      "id": 300410,  
21      "imageType": "jpeg",  
22      "title": "Green Chile Cheeseburgers",  
23      "readyInMinutes": 50,  
24      "servings": 4,  
25      "sourceUrl": "http://www.foodnetwork.com/recipes/guy-fieri/green-chile-cheeseburgers-recipe.html"  
26    }  
27  ],  
28  "nutrients": {  
29    "calories": 2001.17,  
30    "protein": 53.28,  
31    "fat": 153.4,  
32    "carbohydrates": 123.09  
33  }
```

Εικόνα 19: Παράδειγμα ημερήσιου πλάνου (Αποτέλεσμα request)

Μέσω του πίνακα “meals” μας έχουν επιστραφεί τρεις συνταγές για τα τρία βασικά γεύματα της ημέρας, μαζί με μερικές πληροφορίες για την κάθε συνταγή. Τέλος, ο πίνακας “nutrients” μας επιστρέφει πληροφορίες για τα θρεπτικά συστατικά του πλάνου και πιο συγκεκριμένα για τις θερμίδες, την πρωτεΐνη, τα λιπαρά και τους υδατάνθρακες.

Θα ξανακάνουμε το ίδιο request με την διαφορά ότι τώρα θέλουμε να παράγουμε ένα εβδομαδιαίο πλάνο.

```
2  "week": {  
3    "monday": {  
4      "meals": [  
5        {  
6          "id": 1438593,  
7          "imageType": "jpg",  
8          "title": "Tuna pasta bake",  
9          "readyInMinutes": 50,  
10         "servings": 6,  
11         "sourceUrl": "https://www.bbcgoodfood.com/recipes/tuna-pasta-bake"  
12        },  
13        {  
14          "id": 600187,  
15          "imageType": "jpg",  
16          "title": "Crunchy Coconut French Toast",  
17          "readyInMinutes": 25,  
18          "servings": 4,  
19          "sourceUrl": "http://leitesculinaria.com/494/recipes-crunchy-coconut-french-toast.html"  
20        },  
21        {  
22          "id": 702001,  
23          "imageType": "jpg",  
24          "title": "Spicy Cranberry Goat Cheese Pinwheels",  
25          "readyInMinutes": 45,  
26          "servings": 3,  
27          "sourceUrl": "http://www.crumbsandchaos.net/2015/12/spicy-cranberry-goat-cheese-pinwheels/"  
28        }  
29      ],  
30      "nutrients": {  
31        "calories": 2000.73,  
32        "protein": 62.7,  
33        "fat": 97.4,  
34        "carbohydrates": 223.73  
35      }  
36    }  
37  }
```



```
37 "tuesday": {  
38   "meals": [  
39     {  
40       "id": 1697625,  
41       "imageType": "jpg",  
42       "title": "Light and Tasty Tomato Basil Mozzarella Pasta for a Hot Summer Evening",  
43       "readyInMinutes": 25,  
44       "servings": 2,  
45       "sourceUrl": "https://spoonacular.com/light-and-tasty-tomato-basil-mozzarella-pasta-for-a-hot-summer-evening-1697625"  
46     },  
47     {  
48       "id": 21515,  
49       "imageType": "jpg",  
50       "title": "Kimchi and Avocado Quesadillas",  
51       "readyInMinutes": 30,  
52       "servings": 4,  
53       "sourceUrl": "http://www.myrecipes.com/recipe/kimchi-avocado-quesadillas-5040000110984/"  
54     },  
55     {  
56       "id": 587297,  
57       "imageType": "jpg",  
58       "title": "Baked Macaroni and Cheese Casserole",  
59       "readyInMinutes": 45,  
60       "servings": 8,  
61       "sourceUrl": "http://www.merrygourmet.com/2012/02/baked-macaroni-and-cheese-casserole/"  
62     }  
63   ],  
64   "nutrients": {  
65     "calories": 2001.17,  
66     "protein": 73.42,  
67     "fat": 112.24,  
68     "carbohydrates": 177.4  
69   }  
}
```

**Εικόνα 20: Παράδειγμα εβδομαδιαίου πλάνου (Αποτέλεσμα request)**

Παρατηρούμε ότι οι πίνακες έχουν ακριβώς την ίδια δομή με πριν, με την διαφορά ότι τώρα υπάρχουν τρία γεύματα για κάθε μέρα της εβδομάδας.

**Σημείωση:** Έχουν συμπεριληφθεί σε screenshot μόνο τα αποτελέσματα του request που αφορούν τις ημέρες Δευτέρα και Τρίτη αντίστοιχα, για εξοικονόμηση χώρου. Εννοείται ότι την ίδια δομή έχει το αποτέλεσμα και για τις υπόλοιπες μέρες της εβδομάδας.

## 5 Χάρτης

Για την λειτουργία επιλογής κουζίνας με βάση την χώρα στην εφαρμογή μας, έχουμε χρησιμοποιήσει έναν έτοιμο χάρτη τον οποίο μπορείτε να βρείτε στον εξής σύνδεσμο:

<https://codepen.io/team/amcharts/embed/mrmQyB/d6f0cba86f4b823b961a7d5dce535064?default-tab=result&theme-id=light>.

Μέσω αυτού του χάρτη, ο χρήστης μπορεί να επιλέξει κάνοντας κλικ πάνω του την/τις χώρα/χώρες που επιθυμεί και ως αποτέλεσμα θα μας επιστραφεί ο ISO2 κωδικός της/των χώρας/χωρών που έχουν επιλεγεί (λίστα κωδικών χωρών: <https://www.iban.com/country-codes>).

Με αυτόν τον τρόπο, βελτιώνουμε το user interface της εφαρμογής μας, διότι είναι καλύτερα κατανοητό και πιο γρήγορο για τον χρήστη να επιλέξει τις χώρες που θέλει, από το να είχαμε ένα πεδίο στο οποίο θα έγραφε ο χρήστης το όνομα της χώρας που επιθυμεί. Συνεπώς, αποφεύγουμε τον κίνδυνο να υπάρξει κάποιο ορθογραφικό λάθος από την πλευρά του χρήστη, που μετά θα οδηγούσε σε σφάλμα και διακοπή της λειτουργίας της εφαρμογής.

Ακολουθεί screenshot επίδειξης του χάρτη.

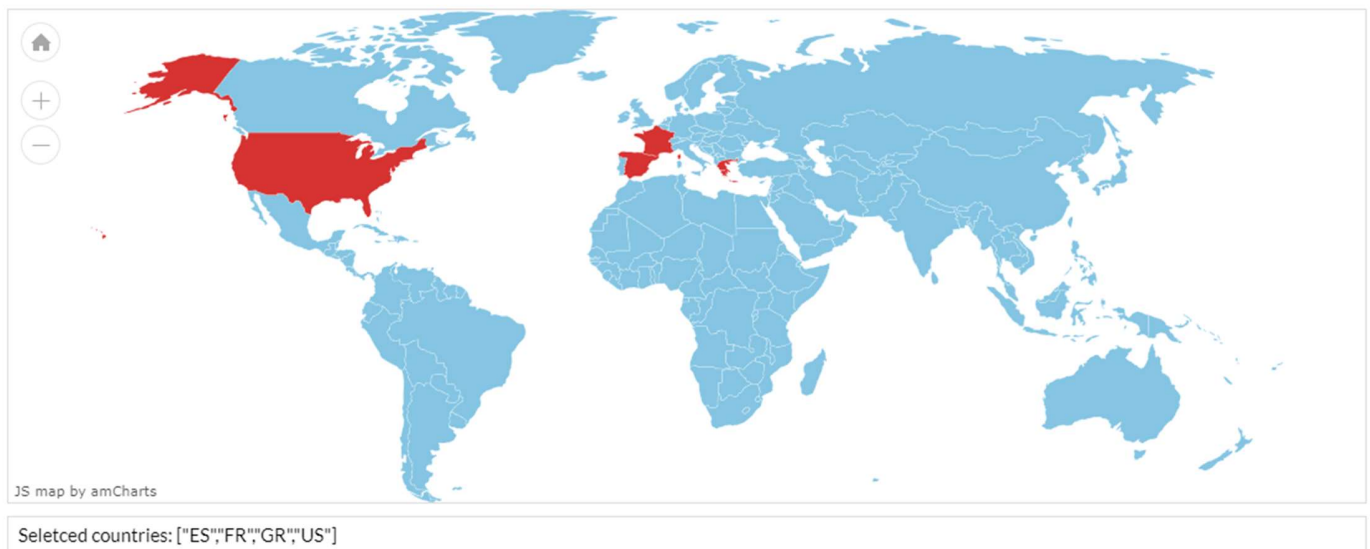


Click countries to select



Εικόνα 21: Χάρτης

Click countries to select



Εικόνα 22: Χάρτης με επιλεγμένες χώρες

## 5.1 Υποστηριζόμενες κουζίνες από το Spoonacular API

Σύμφωνα με το documentation του API, οι υποστηριζόμενες [κουζίνες](#) είναι οι εξής:

### Cuisines

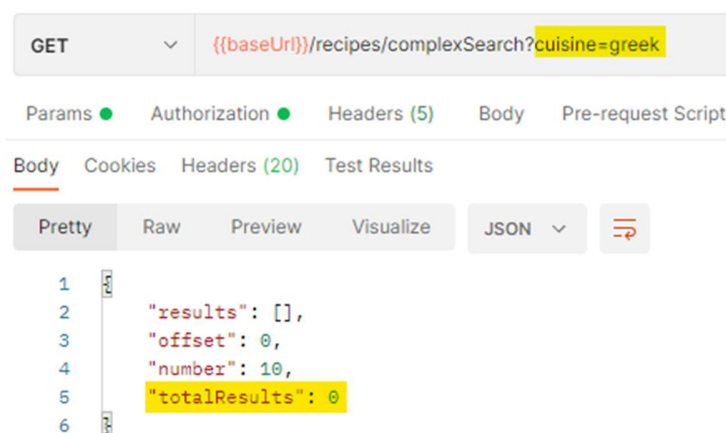
Every API endpoint asking for a `cuisine` parameter can be fed with any of these cuisines.

- African
- American
- British
- Cajun
- Caribbean
- Chinese
- Eastern European
- European
- French
- German
- Greek
- Indian
- Irish
- Italian
- Japanese
- Jewish
- Korean
- Latin American
- Mediterranean
- Mexican
- Middle Eastern
- Nordic
- Southern
- Spanish
- Thai
- Vietnamese

Εικόνα 23: Υποστηριζόμενες κουζίνες του Spoonacular API

### Σημαντική σημείωση:

Κατά την διάρκεια ανάπτυξης της εφαρμογής, το API επέστρεφε κανονικά αποτελέσματα που αφορούν την ελληνική κουζίνα, ωστόσο το τελευταίο διάστημα το API δεν επιστρέφει κανένα αποτέλεσμα παρόλο που στο documentation τους, ακόμα υπάρχει στην λίστα των χωρών που υποστηρίζονται. Εμείς έχουμε μεριμνήσει να εμφανίζεται κάποιο μήνυμα στον χρήστη όταν επιλέγει μια χώρα που να μην υποστηρίζεται από το API, ωστόσο δεν έχουμε συμπεριλάβει την Ελλάδα ως μη υποστηριζόμενη χώρα, αφού όσο αναπτύσσαμε την εφαρμογή λειτουργούσε κανονικά. Ευελπιστούμε ότι το πρόβλημα αυτό κάποια στιγμή θα λυθεί από την πλευρά του API και έτσι θα λειτουργεί κανονικά και η εφαρμογή μας. Ακολουθεί πρόσφατο screenshot που αποδεικνύει ότι το API δεν επιστρέφει αποτέλεσμα.



Εικόνα 24: Κανένα αποτέλεσμα για ελληνική κουζίνα

## 5.2 Επεξήγηση κώδικα της εφαρμογής

Στον κώδικά μας επεξεργαζόμαστε κατάλληλα την πληροφορία που αντλούμε μέσω του χάρτη, για να πραγματοποιήσουμε το αντίστοιχο JSON request στο API.

Αρχικά, έχουμε δημιουργήσει ένα dictionary με τις αντιστοιχίες των κωδικών των χωρών με την αντίστοιχη κουζίνα. Με αυτό τον τρόπο, παίρνουμε τους κωδικούς των χωρών που έχει επιλέξει ο χρήστης από τον χάρτη και μπορούμε να βρούμε την κουζίνα που αντιστοιχεί στις χώρες αυτές, ώστε να πραγματοποιήσουμε το JSON request. Στο dictionary υπάρχουν μόνο οι χώρες που οι κουζίνες τους υποστηρίζονται από το API.

```
// Creating a dictionary using collection-initializer syntax
Dictionary<string, string> cuisines = new Dictionary<string, string>
{
    {"AO", "african"}, {"BF", "african"}, {"BI", "african"}, {"BJ", "african"}, {"BW", "african"}, {"CD", "african"}, {"CF", "african"}, {"CG", "african"},
    {"CI", "african"}, {"CM", "african"}, {"DJ", "african"}, {"DZ", "african"}, {"EH", "african"}, {"ER", "african"}, {"ET", "african"}, {"GA", "african"},
    {"GH", "african"}, {"GM", "african"}, {"GN", "african"}, {"GQ", "african"}, {"GW", "african"}, {"KE", "african"}, {"LR", "african"}, {"LS", "african"},
    {"LY", "african"}, {"MA", "african"}, {"MG", "african"}, {"ML", "african"}, {"MR", "african"}, {"MW", "african"}, {"MZ", "african"}, {"NA", "african"},
    {"NE", "african"}, {"NG", "african"}, {"RW", "african"}, {"SD", "african"}, {"SL", "african"}, {"SN", "african"}, {"SO", "african"}, {"SS", "african"},
    {"SZ", "african"}, {"TD", "african"}, {"TG", "african"}, {"TN", "african"}, {"TZ", "african"}, {"UG", "african"}, {"ZA", "african"}, {"ZM", "african"},
    {"ZW", "african"}, {"BS", "caribbean"}, {"CU", "caribbean"}, {"DO", "caribbean"}, {"HT", "caribbean"}, {"JM", "caribbean"}, {"PR", "caribbean"},
    {"AR", "latin american"}, {"BO", "latin american"}, {"BR", "latin american"}, {"BZ", "latin american"}, {"CL", "latin american"}, {"CO", "latin american"},
    {"CR", "latin american"}, {"EC", "latin american"}, {"FK", "latin american"}, {"GF", "latin american"}, {"GT", "latin american"}, {"GY", "latin american"},
    {"HN", "latin american"}, {"NI", "latin american"}, {"PA", "latin american"}, {"PE", "latin american"}, {"PY", "latin american"}, {"SR", "latin american"},
    {"SV", "latin american"}, {"TT", "latin american"}, {"UY", "latin american"}, {"VE", "latin american"}, {"AE", "middle eastern"}, {"EG", "middle eastern"},
    {"IL", "middle eastern"}, {"IQ", "middle eastern"}, {"IR", "middle eastern"}, {"JO", "middle eastern"}, {"KW", "middle eastern"}, {"LB", "middle eastern"},
    {"OM", "middle eastern"}, {"PS", "middle eastern"}, {"QA", "middle eastern"}, {"SA", "middle eastern"}, {"SY", "middle eastern"}, {"TR", "middle eastern"},
    {"YE", "middle eastern"}, {"DK", "nordic"}, {"EE", "nordic"}, {"FI", "nordic"}, {"IS", "nordic"}, {"LT", "nordic"}, {"LV", "nordic"}, {"NO", "nordic"},
    {"SE", "nordic"}, {"CN", "chinese"}, {"DE", "german"}, {"ES", "spanish"}, {"FR", "french"}, {"GB", "british"}, {"GR", "greek"}, {"IE", "irish"}, {"IN", "indian"},
    {"IT", "italian"}, {"JP", "japanese"}, {"KR", "korean"}, {"MX", "mexican"}, {"US", "american"}, {"VN", "vietnamese"}
};
```

Εικόνα 25: Η δομή του dictionary

Στην συνέχεια, όταν ο χρήστης πατήσει το κουμπί για αναζήτηση των συνταγών, τρέχει στον server η μέθοδος GetCuisines(), η οποία δημιουργεί ένα string με τα ονόματα των κουζινών που θα περαστεί στο url του request.

Αρχικά, υπάρχει έλεγχος για το εάν ο χρήστης έχει πραγματοποιήσει αναζήτηση συνταγών χωρίς να έχει επιλέξει καμία χώρα και του εμφανίζεται κατάλληλο μήνυμα λάθους.

Αν ο χρήστης έχει επιλέξει κάποια ή κάποιες χώρες, δημιουργείται η λίστα result που περιέχει τις επιλεγμένες χώρες και διατρέχοντας τα στοιχεία τις, ελέγχουμε αν η κάθε χώρα της λίστας υπάρχει στο dictionary.

Αν υπάρχει τότε προστίθεται στο τελικό string με όνομα final, αλλιώς δεν προστίθεται. Στην συνέχεια ελέγχεται αν το final είναι null.

Αν είναι null τότε σημαίνει ότι ο χρήστης έχει επιλέξει χώρες που δεν υποστηρίζονται και του εμφανίζεται κατάλληλο μήνυμα λάθους.

Αλλιώς όλα έχουν πάει καλά και ο χρήστης μεταφέρεται στην επόμενη σελίδα όπου εκτελείται και το request, για να δει τα αποτελέσματα που θα του εμφανιστούν.

```
protected void GetCuisines(object sender, EventArgs e)
{
    string s = hdnText.Value;
    if (s == "-") // No country selected
    {
        Response.Write("<script>alert('No country selected.')
```

Εικόνα 26: Κώδικας της GetCuisines()

```
$.getJSON({
    url: "https://api.spoonacular.com/recipes/complexSearch?apiKey=a94520b4cba649cab8ce0dcf4abfa3f6&cuisine="
    + document.getElementById('<%= result.ClientID %>').innerHTML + "&addRecipeInformation=true&addRecipeNutrition=true&number=18",
    success: function (res) {
        // Set the contents of the cards in card view
        res.results.forEach((element) => {
            var t = document.getElementById("cards");
            var r = document.createElement('div');
            r.className = "card";
            r.innerHTML = `
                <div class="column">
                
                <h4 id="titles">${element.title}</h4>
                <p id="info" class="title">${element.nutrition.nutrients[0].amount} kcal, ready in ${element.readyInMinutes}</p>
                <div style="margin: 24px 0;">
                <a href="SearchCuisine.aspx?result=${rsIt}&id=${element.id}" runat="server"><i class="fa fa-heart"></i></a>
                </div>
                <p><button onclick="window.open('${element.spoonacularSourceUrl}', '_blank')">View More</button></p>
                </div>
            `;
            t.appendChild(r);
        });
    }
});
```

Εικόνα 27: Κώδικας του request και ενημέρωση περιεχομένου σελίδας



## 6 Αποθήκευση πλάνου και αγαπημένων συνταγών

Μερικές από τις βασικές λειτουργίες της εφαρμογής μας είναι η δημιουργία και αποθήκευση ημερήσιου και εβδομαδιαίου πλάνου γευμάτων, καθώς και η αποθήκευση αγαπημένων συνταγών. Παρακάτω αναλύεται ο κώδικας που υλοποιεί αυτές τις λειτουργίες.

### 6.1 Επεξήγηση κώδικα αποθήκευσης πλάνων

Έχουμε ήδη αναλύσει το πως γίνεται η δημιουργία εβδομαδιαίων και ημερήσιων πλάνων μέσω JSON request. Για αυτό τον λόγο, θα επικεντρωθούμε στην εξήγηση της διαδικασίας της αποθήκευσης των πλάνων αυτών.

Όταν ο χρήστης πατήσει το κουμπί για αποθήκευση ενός πλάνου, είτε εβδομαδιαίου είτε ημερήσιου, καλείται η μέθοδος SaveMeal() που τρέχει στον server.

Αρχικά, δημιουργείται μια λίστα αντικειμένων Meals, η οποία θα αποθηκεύει όλα τα αντικείμενα meals που θα δημιουργηθούν με τα αντίστοιχα χαρακτηριστικά τους. Τα περιεχόμενα αυτής της λίστας είναι αυτά που θα αποθηκευτούν τελικά και στην βάση. Στην συνέχεια διαβάζουμε τα περιεχόμενα των πινάκων της σελίδας. Είναι ο πίνακας meals που περιέχει τα γεύματα για την κάθε μέρα και ο πίνακας nutrients που περιέχει τα θρεπτικά συστατικά του πλάνου. Με βάση τα δεδομένα των πινάκων, δημιουργείται κάθε φορά αντίστοιχο αντικείμενο Meals το οποίο και μετά θα εισαχθεί στην λίστα. Μόλις διαβαστούν όλα τα δεδομένα από τους πίνακες και γεμίσει η λίστα αντικειμένων Meals, τότε εκτελείται το query προς την βάση. Αν δεν υπάρχει προηγούμενη εγγραφή στην βάση για τον συγκεκριμένο χρήστη τότε εκτελείται μια insert εντολή, αλλιώς αν ήδη υπάρχει εγγραφή εκτελείται ένα update. Αυτό συμβαίνει, διότι κάθε φορά αποθηκεύουμε το τελευταίο πλάνο που έχει δημιουργήσει ο χρήστης.

```
protected void SaveMeal(object sender, EventArgs e)
{
    // List of Meals objects contents of which will be saved to database
    List<Meals> mealsdb = new List<Meals>();
    // Secondary auxiliary list
    List<List<string>> axlist = new List<List<string>>();

    // Get json string containing the values of second table (meals)
    var json = meals.Value;

    var objects = JObject.Parse(json); // parse as array
    foreach (JObject root in objects)
    {
        // Save the Day and Id of the second table to the auxiliary list (3 recipes for each day)
        List<string> flag = new List<string>();
        flag.Add((string)root.GetValue("Day"));
        flag.Add((string)root.GetValue("Id"));
        axlist.Add(flag);
    }

    // Get json string containing the values of first table (nutrients)
    var json1 = nutrients.Value;

    var objects1 = JObject.Parse(json1); // parse as array
    foreach (JObject root in objects1)
    {
        // List containing the ids of the 3 recipes for each day
        List<string> ids = new List<string>();
        foreach (List<string> sublist in axlist)
        {
            // If days match
            if (((string)root.GetValue("Day")).Equals(sublist[0]))
            {
                // Add the id
                ids.Add(sublist[1]);
            }
        }
    }
}
```

```
// Create the Meals object which is going to be saved in database and add it to the list
Meals mobject = new Meals((string)HttpContext.Current.Session["Username"],
    (string)root.GetValue("Day"), ids[0], ids[1], ids[2], (decimal)root.GetValue("Calories"),
    (decimal)root.GetValue("Protein"), (decimal)root.GetValue("Fat"), (decimal)root.GetValue("Carbohydrates"));
mealsdb.Add(mobject);
}

int rowsAffected = -1; // false value
// Insert or overwrite data in database
foreach (Meals m in mealsdb)
{
    string query = "insert into mealplans values (@username, @days, @id1, @id2, @id3, @calories, @protein, @fat, @carbohydrates)" +
        "on conflict (username, days) do update set id1 = @id1, id2 = @id2, id3 = @id3, calories = @calories, protein = @protein, " +
        "fat = @fat, carbohydrates = @carbohydrates";

    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        //define query's parameters
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        command.Parameters.AddWithValue("username", m.username);
        command.Parameters.AddWithValue("days", m.days);
        command.Parameters.AddWithValue("id1", m.id1);
        command.Parameters.AddWithValue("id2", m.id2);
        command.Parameters.AddWithValue("id3", m.id3);
        command.Parameters.AddWithValue("calories", m.calories);
        command.Parameters.AddWithValue("protein", m.protein);
        command.Parameters.AddWithValue("fat", m.fat);
        command.Parameters.AddWithValue("carbohydrates", m.carbohydrates);
        rowsAffected = command.ExecuteNonQuery(); //run query
        connection.Close();
    }
    catch { break; }
}

if (rowsAffected == -1)
{
    string notification1 = "Something went wrong, please try again later.";
    Response.Write("<script>alert('" + notification1 + "') ; </script>");
}
}
```

```
if (rowsAffected == -1)
{
    string notification1 = "Something went wrong, please try again later.";
    Response.Write("<script>alert('" + notification1 + "') ; </script>");
}
else
{
    string notification1 = "Meal plan created successfully!";
    Response.Write("<script>alert('" + notification1 + "') ; </script>");
}
```

Εικόνα 28: Κώδικας της SaveMeal()



## 6.2 Επεξήγηση κώδικα αποθήκευσης αγαπημένων συνταγών

Όταν ο χρήστης πατήσει το κουμπί για αποθήκευση μιας συνταγής, καλείται η μέθοδος PageLoad() που τρέχει στον server.

Αρχικά, παίρνουμε το id της συνταγής την οποία θέλει να αποθηκεύσει και στην συνέχεια ελέγχουμε αν ο χρήστης είναι συνδεδεμένος στον λογαριασμό του. Στην συνέχεια εκτελείται ένα insert query στην βάση, εισάγοντας στον πίνακα favorites μια εγγραφή με το username του χρήστη και το id της συνταγής που έχει κάνει αγαπημένη.

```
protected void Page_Load(object sender, EventArgs e)
{
    string s = Request.QueryString["id"];
    System.Diagnostics.Debug.WriteLine(s);

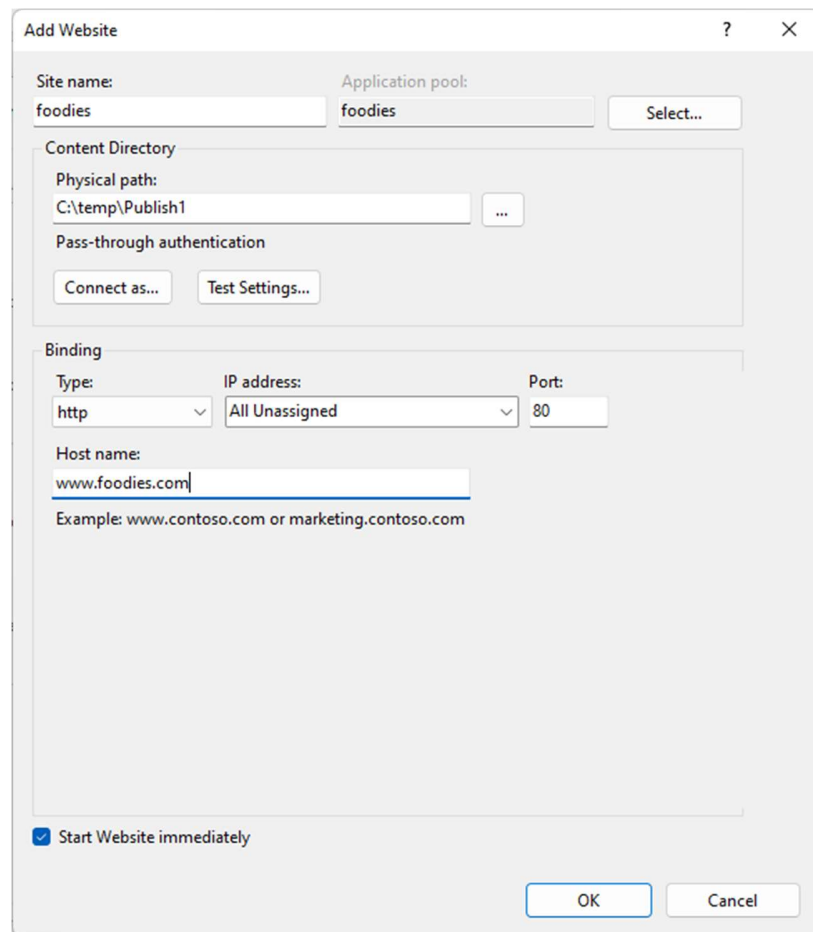
    if (s != null)
    {
        // User is not supposed to use this webform, redirect to Login
        if (Session.Contents.Count == 0)
        {
            Response.Redirect("Login.aspx");
        }
        else
        {
            // Add recipe to favorites
            string query = "insert into favorites values (@username, @id)";
            int rowsAffected = -1; //false value
            try
            {
                NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
                connection.Open();
                //define query's parameters
                NpgsqlCommand command = new NpgsqlCommand(query, connection);
                command.Parameters.AddWithValue("username", (string)HttpContext.Current.Session["Username"]);
                command.Parameters.AddWithValue("id", s);
                rowsAffected = command.ExecuteNonQuery(); //run query
                connection.Close();
            }
            catch{}
            if (rowsAffected == -1)
            {
                string notification1 = "You have already added this recipe to your favorites.";
                Response.Write("<script>alert('" + notification1 + "') ; </script>");
            }
            else
            {
                string notification1 = "Recipe added to your favorites successfully!";
                Response.Write("<script>alert('" + notification1 + "') ; </script>");
            }
        }
    }
}
```

Εικόνα 29: Κώδικας της PageLoad()

## 7 Εισαγωγή της ιστοσελίδας στον IIS και δημιουργία πιστοποιητικού

### 7.1 Προσθήκη της εφαρμογής στον IIS

Καταρχάς, για να εγκαταστήσουμε την εφαρμογή μας στον IIS την κάναμε Publish μέσα από το περιβάλλον του Visual Studio. Στη συνέχεια, κάναμε προσθήκη της ιστοσελίδας μας μέσω του IIS Manager με τις παρακάτω ρυθμίσεις.



Εικόνα 30: Εισαγωγή σελίδας στον IIS

Με σκοπό να πληκτρολογούμε για την είσοδο στην εφαρμογή το domain name που ορίσαμε αντί για «localhost», τροποποιήσαμε το αρχείο hosts που βρίσκεται στο path C:\Windows\System32\drivers\etc\, όπως φαίνεται στο παρακάτω screenshot.

```
# localhost name resolution is handled within DNS itself.
# 127.0.0.1      localhost
# ::1           localhost

127.0.0.1      www.foodies.com
```

Εικόνα 31: Αρχείο hosts

## 7.2 Δημιουργία SSL certificate

Για την δημιουργία ενός SSL πιστοποιητικού τρέξαμε την παρακάτω εντολή στο PowerShell με δικαιώματα διαχειριστή.

```
Administrator: Windows PowerShell (x86)
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> New-SelfSignedCertificate -DnsName "www.foodies.com" -CertStoreLocation "cert:\LocalMachine\My"

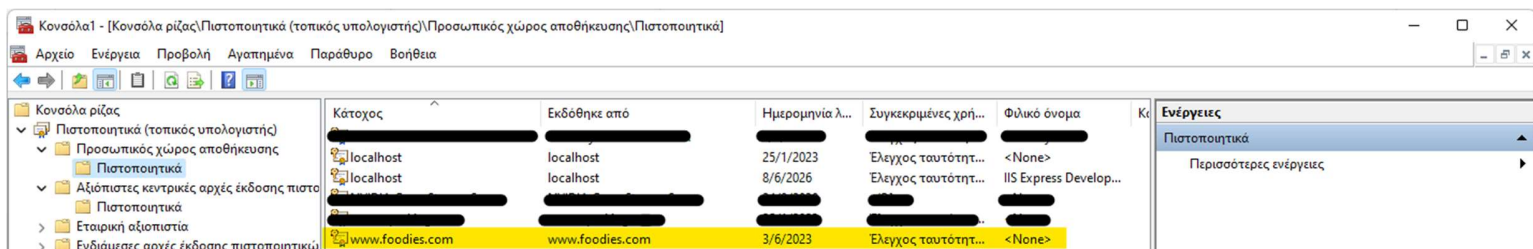
PSParentPath: Microsoft.PowerShell.Security\Certificate::LocalMachine\My

Thumbprint                               Subject
-----
ECFD2BFD6A331C37309FAEA0216049E03340B9C9  CN=www.foodies.com

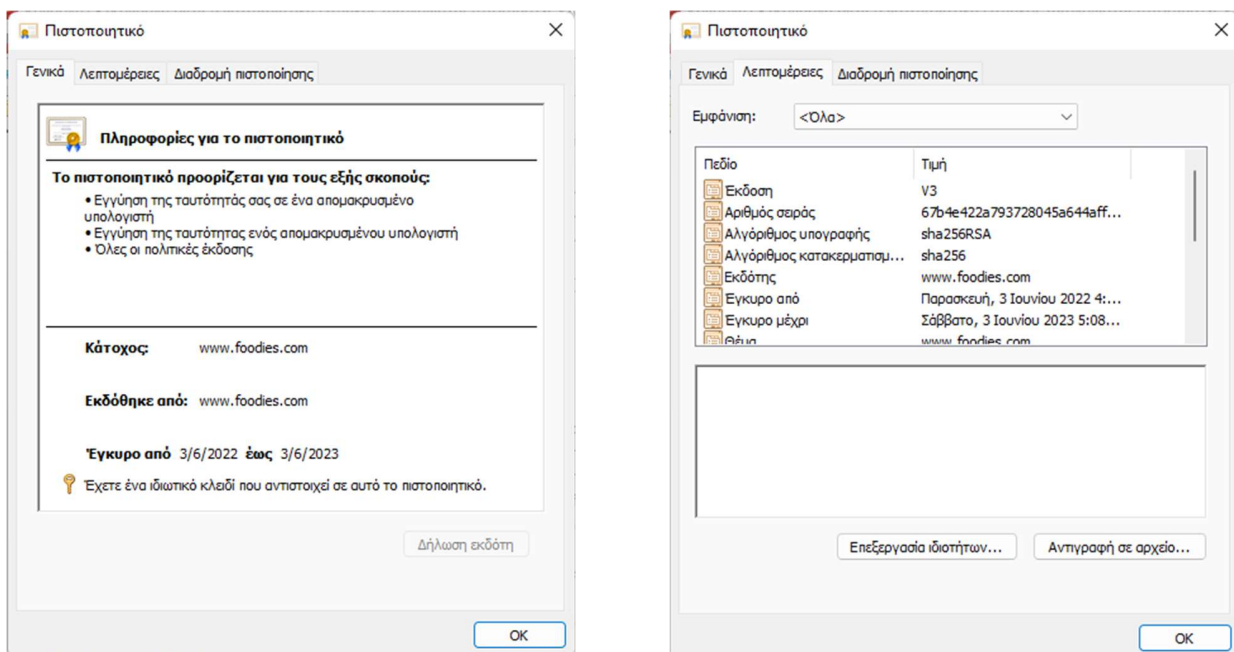
PS C:\WINDOWS\system32>
```

Εικόνα 32: Δημιουργία SSL certificate

Το αποτέλεσμα της εντολής ήταν η δημιουργία ενός αυτό-υπογεγραμμένου πιστοποιητικού, το οποίο μπορούμε να το εντοπίσουμε μέσω της εφαρμογής mmc.exe. Πιο συγκεκριμένα βρίσκεται στον υπο-φάκελο «Πιστοποιητικά», του φακέλου «Προσωπικός χώρος αποθήκευσης».

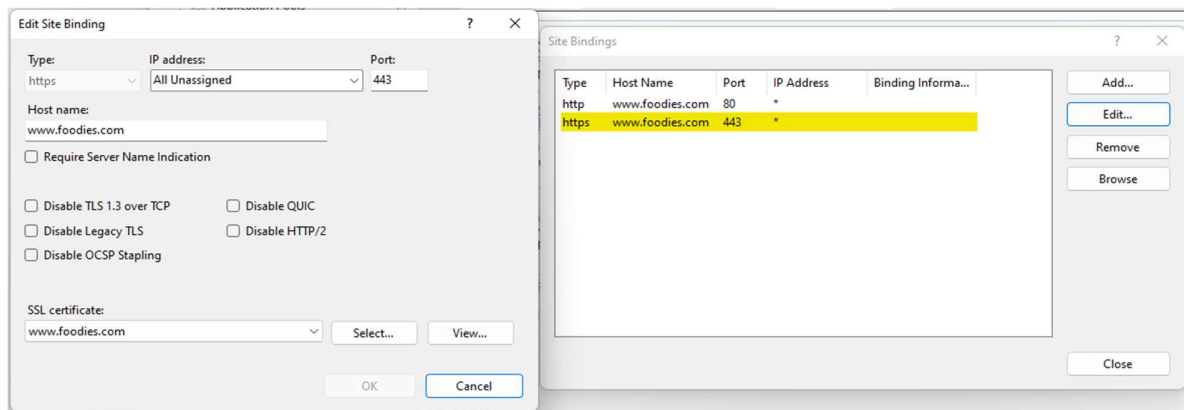


Εικόνα 33: Προσωπικός χώρος αποθήκευσης

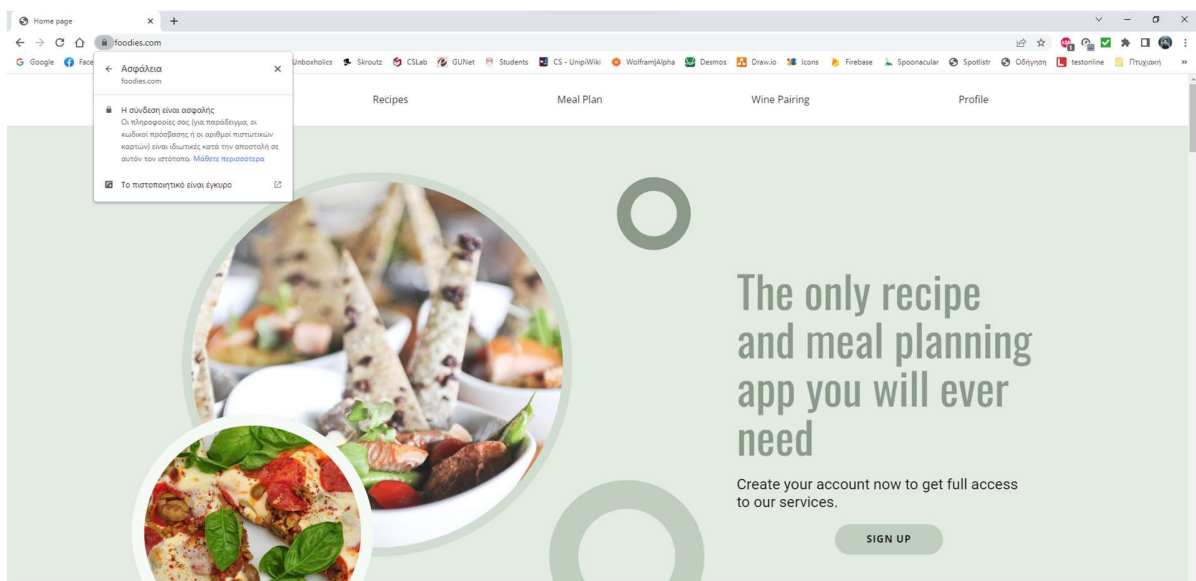


Εικόνα 34: Πιστοποιητικό

Το τελικό βήμα είναι να προσθέσουμε ένα https binding στην ιστοσελίδα μας μέσω του IIS.



Το αποτέλεσμα όλων των παραπάνω είναι όταν πληκτρολογούμε στον browser την διεύθυνση [www.foodies.com](http://www.foodies.com), να συνδεόμαστε στην ιστοσελίδα μας με ασφάλεια, όπως φαίνεται παρακάτω.



---

24

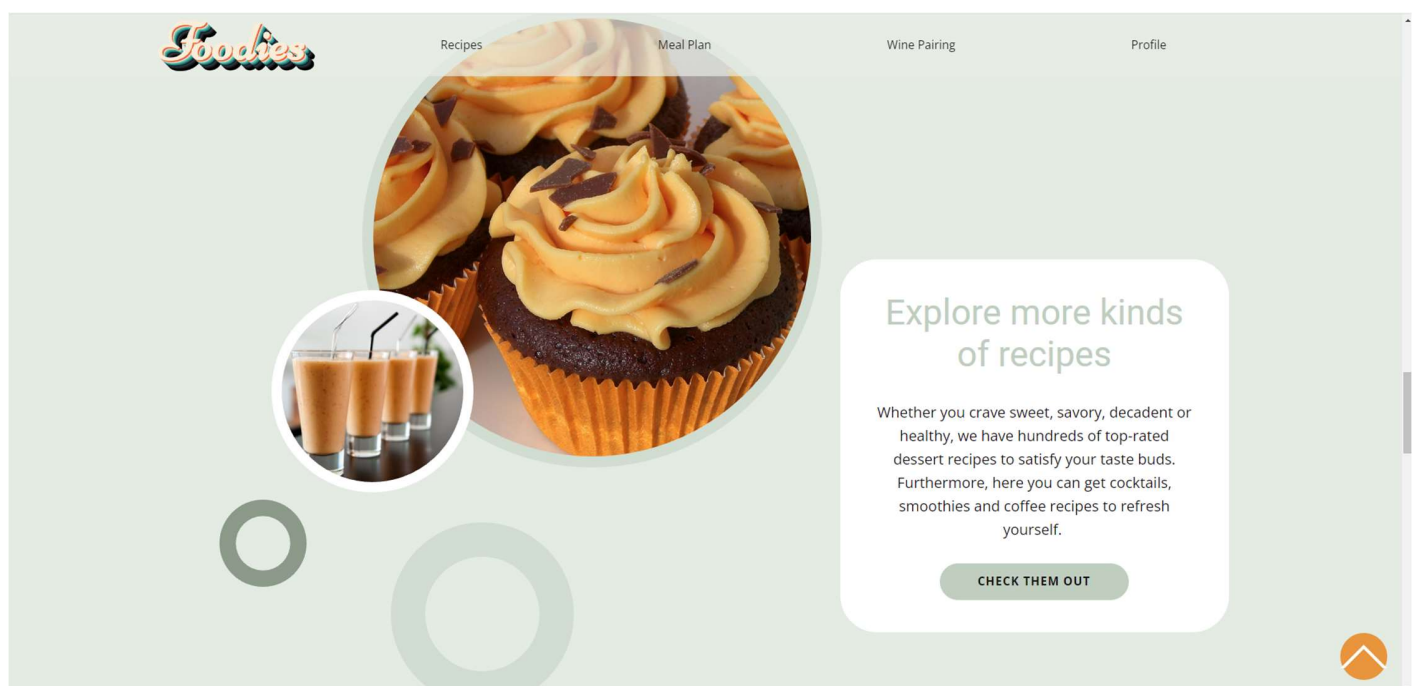
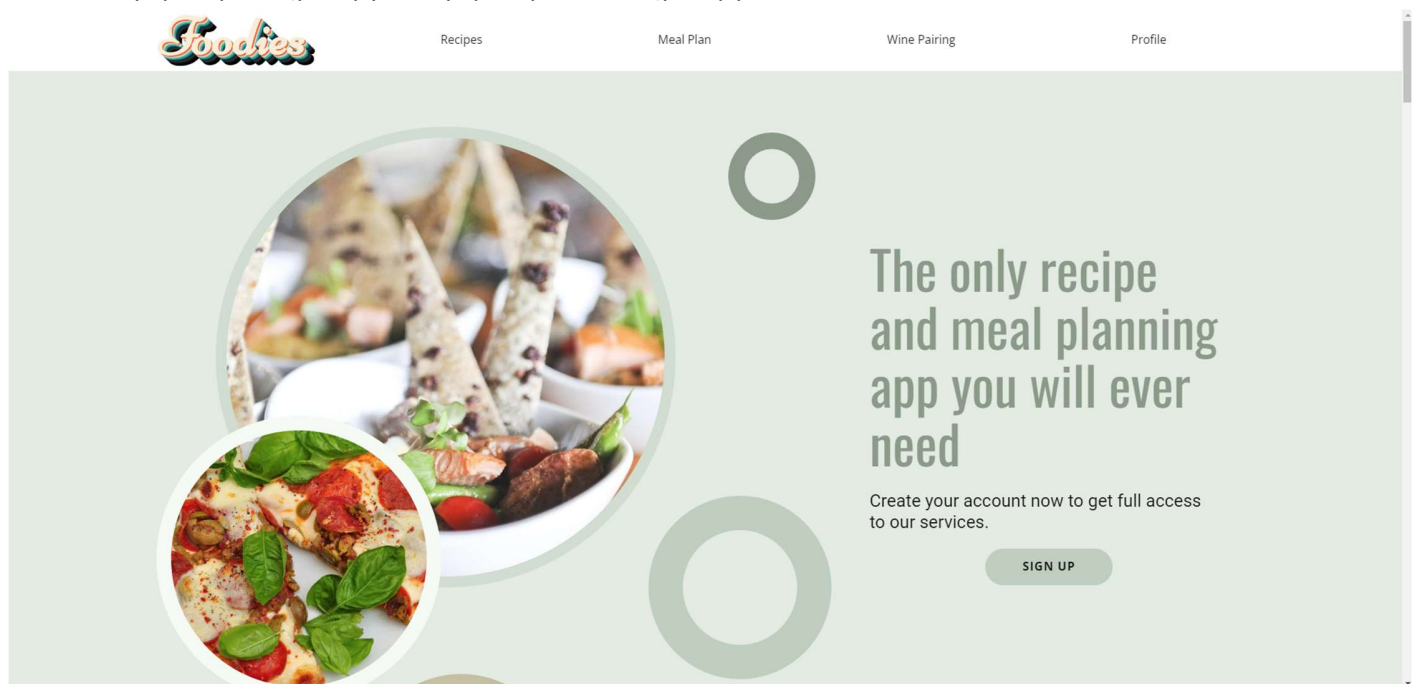


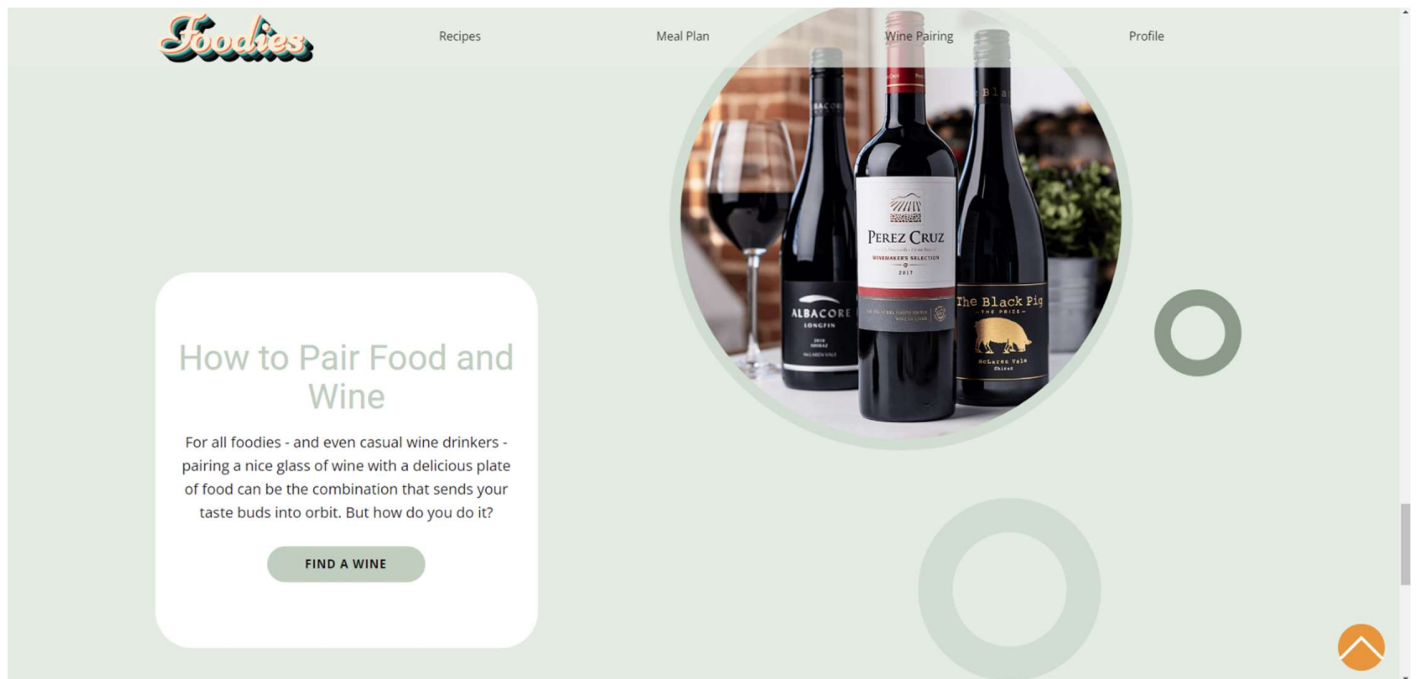
## 8 Εγχειρίδιο χρήστη

### 8.1 Σύνοψη παρουσίαση του προγράμματος

- Home – Main Page

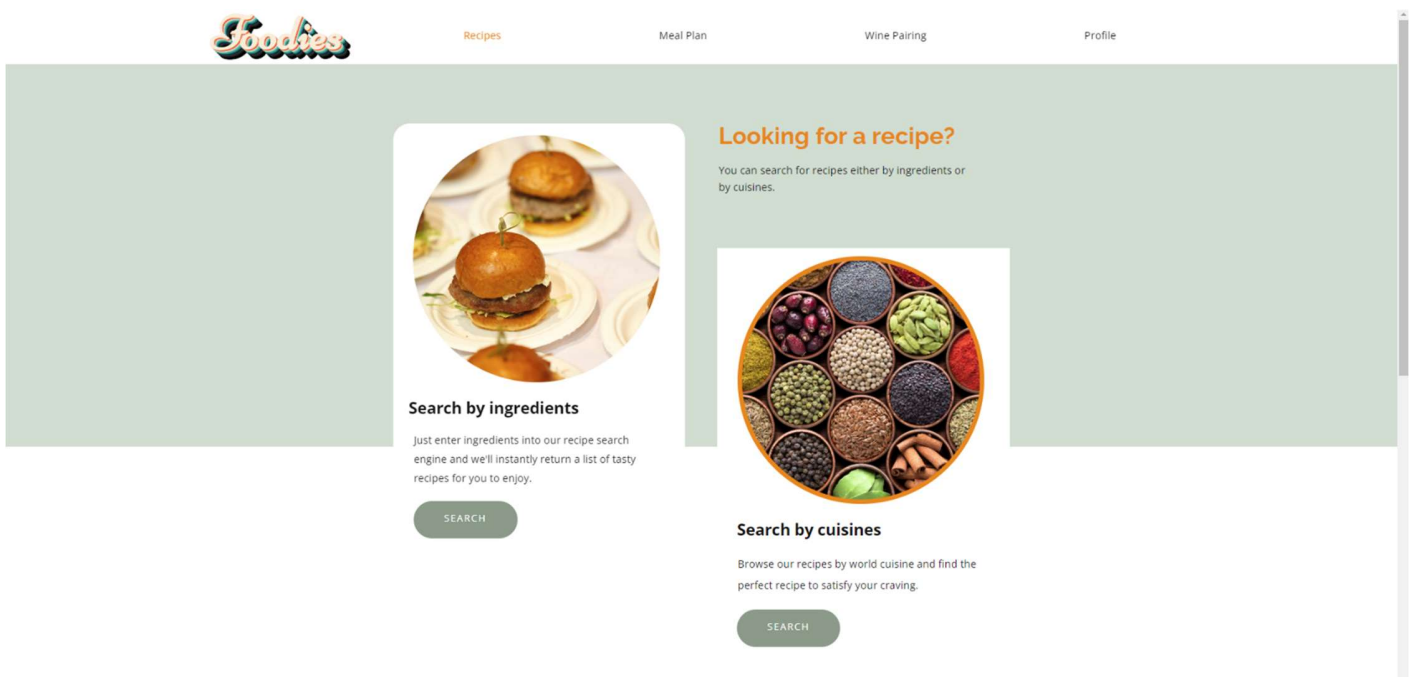
Η main page μας αποτελείται από όλες τις βασικές επιλογές που μπορεί να χρειαστεί ένας πιθανός χρήστης της εφαρμογής. Υπάρχουν ανακατευθύνσεις για τις σελίδες αναζήτησης συνταγών, κρασιού, σύνδεση στον λογαριασμό, δημιουργία λογαριασμού και δημιουργία πλάνου.



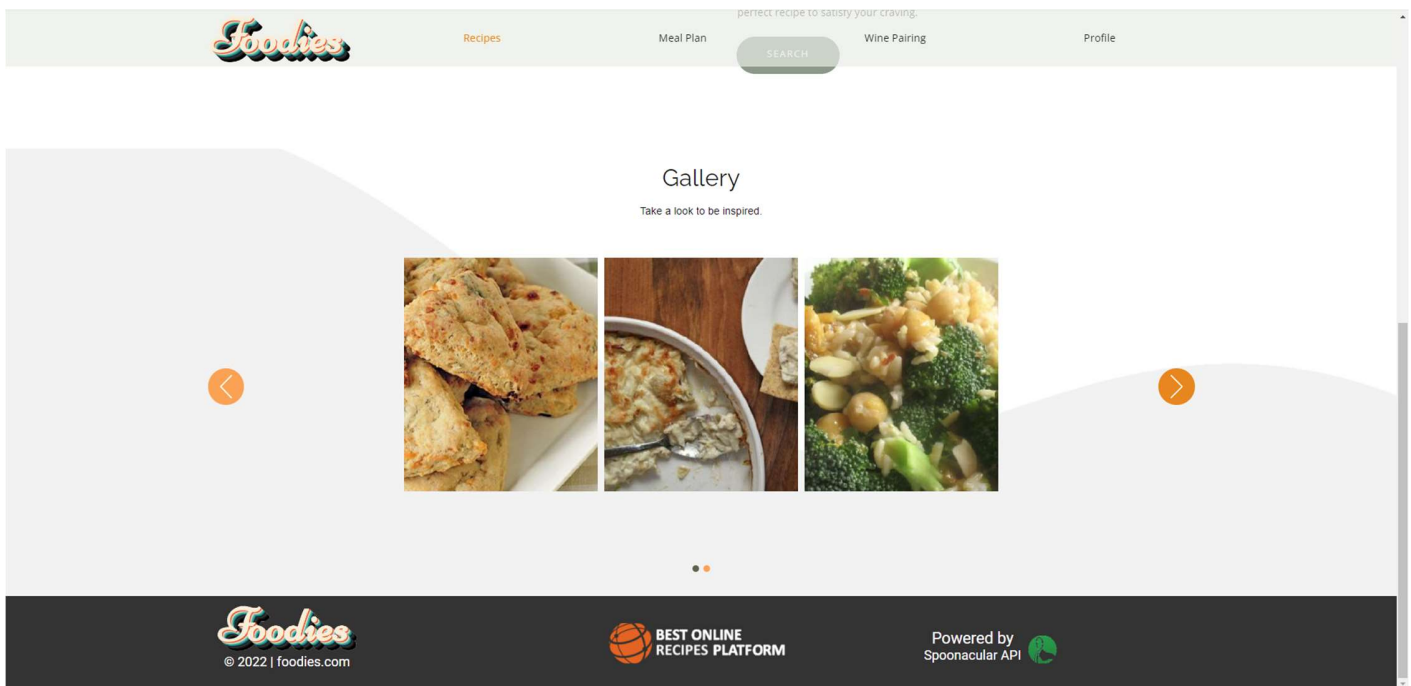


- Recipes

Εδώ ο χρήστης μπορεί να επιλέξει αν θέλει να αναζητήσει συνταγές με βάση κάποιο κύριο συστατικό ή με βάση κάποια κουζίνα. Ανάλογα με την επιλογή του θα ανακατευθυνθεί σε άλλη αντίστοιχη σελίδα. Τέλος, υπάρχει ένα slideshow που εμφανίζονται κάθε φορά τυχαίες φωτογραφίες από συνταγές που υπάρχουν στην σελίδα μας για να εμπνευστεί.

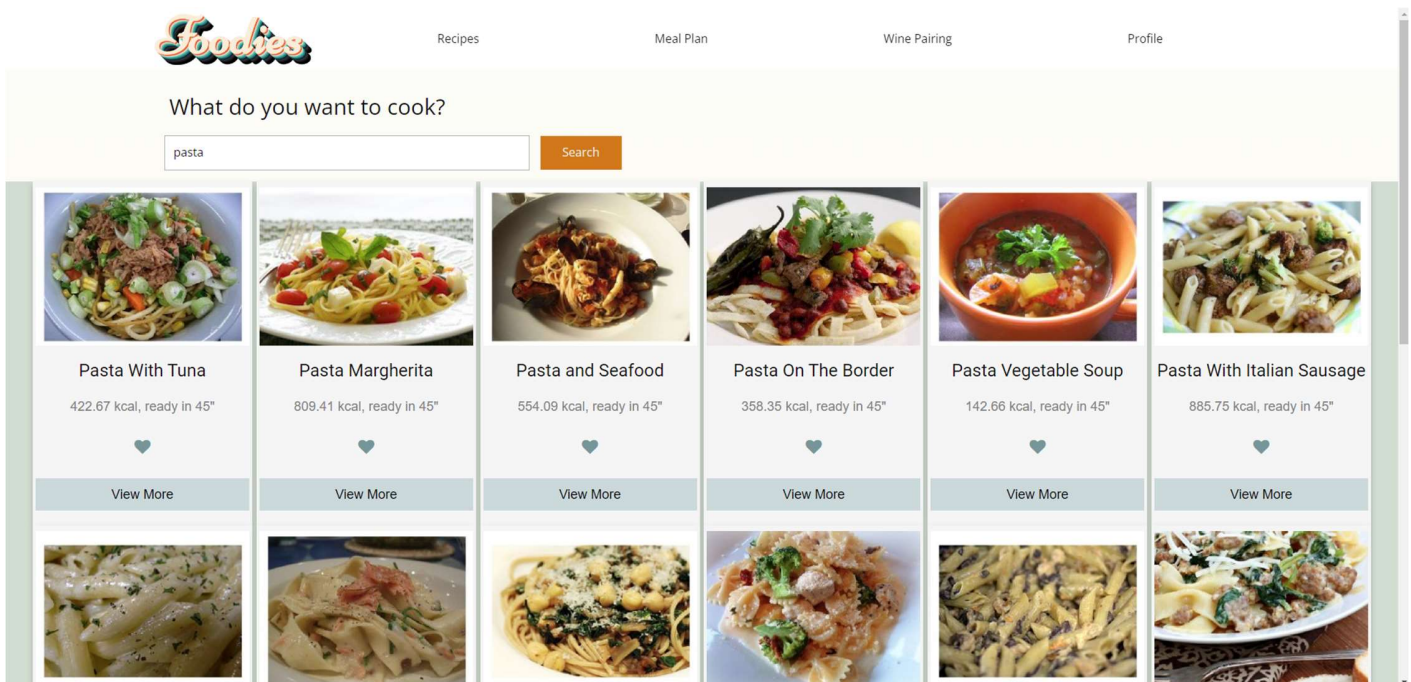






- SearchIngredient

Σε αυτή την σελίδα ο χρήστης μπορεί να πληκτρολογήσει ένα ή περισσότερα βασικά συστατικά που θέλει να έχουν οι συνταγές που θα του εμφανιστούν. Μετά την εμφάνιση των αποτελεσμάτων μπορεί να πατήσει το κουμπί View More για περισσότερες πληροφορίες της συνταγής, καθώς και να την προσθέσει στα αγαπημένα του πατώντας την καρδούλα. Η λειτουργία προσθήκης στα αγαπημένα υπάρχει μόνο για τους εγγεγραμμένους χρήστες της εφαρμογής.



- SearchMap

Εδώ ο χρήστης μπορεί να επιλέξει χώρες από τον χάρτη, ώστε να του εμφανιστούν συνταγές από τις κουζίνες των χωρών αυτών. Στην συνέχεια, κάνοντας κλικ στο κουμπί “Enter” θα μεταφερθεί στην σελίδα SearchCuisine όπου και θα του εμφανιστούν οι συνταγές.


RecipesMeal PlanWine PairingProfile


### Food Around the World Map with Recipes to Explore

From Asian cuisine to American dishes, the maps offer an in-depth look at various cuisines from all around the globe. It showcases not only the age-old classics known to all but smaller and less-known delicacies from countries that aren't in the spotlight (yet might have hidden gems).

Scroll down below to explore the world atlas of taste and see what it has to offer or try navigating their food map yourself here.

Click on the countries you want to select, using the map below.






JS map by amCharts

Selected countries: ["FR","GB","US"]

ENTER

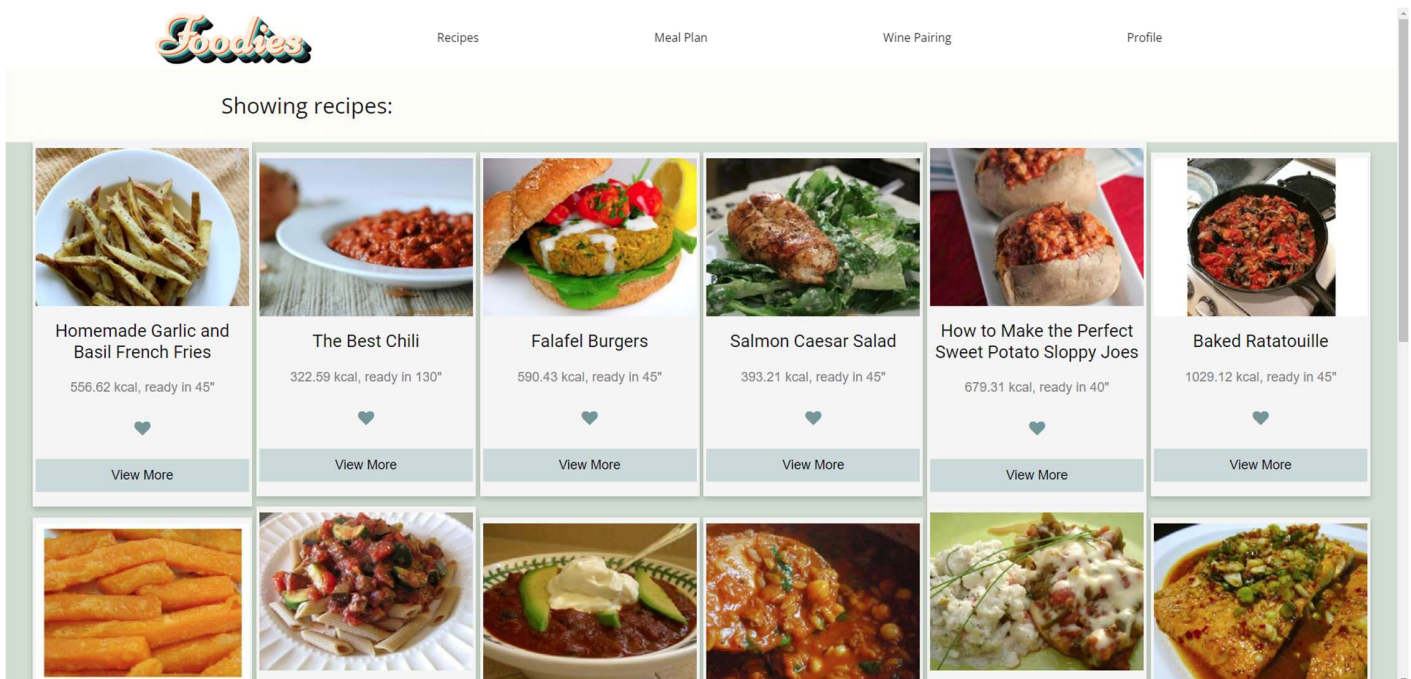


© 2022 | foodies.com



Powered by Spoonacular API 

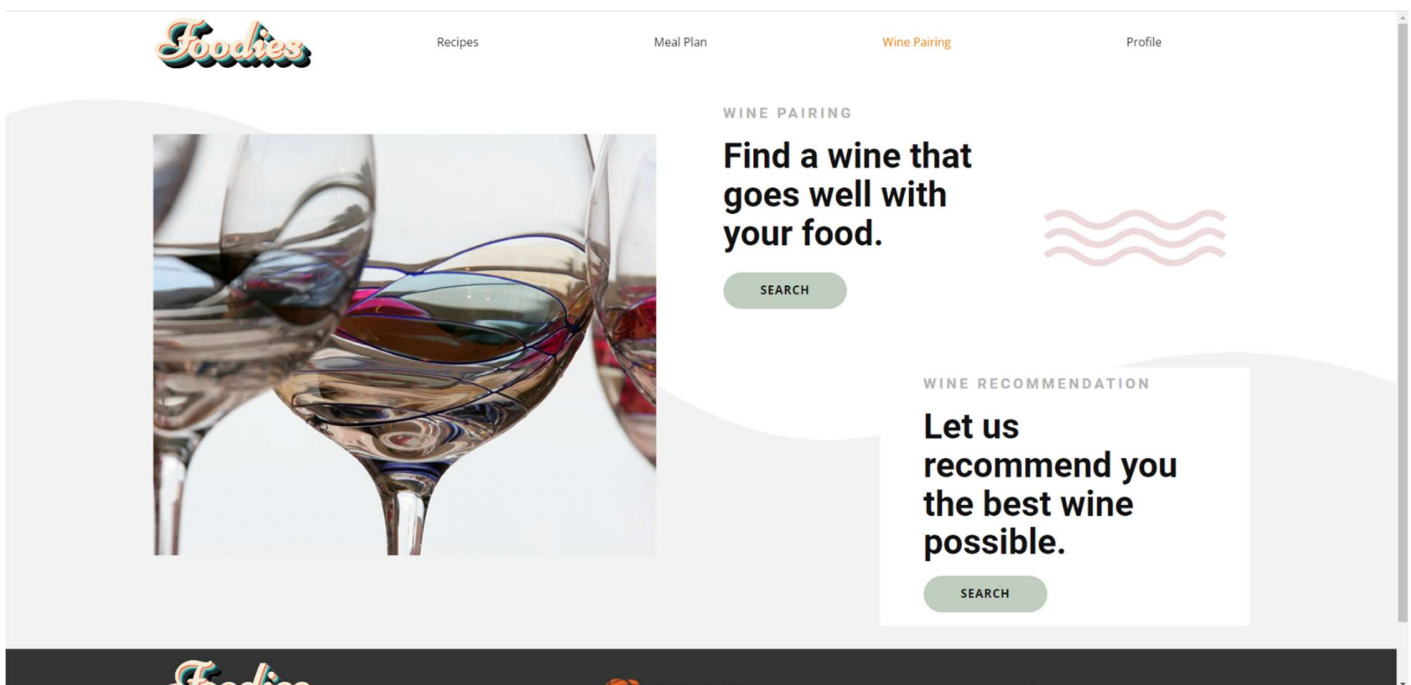
- SearchCuisine



The screenshot shows the SearchCuisine website interface. At the top, there is a navigation bar with the 'Foodies' logo and links for 'Recipes', 'Meal Plan', 'Wine Pairing', and 'Profile'. Below the navigation bar, a section titled 'Showing recipes:' displays a grid of six recipe cards. Each card features a food image, the recipe name, the number of calories, the preparation time, a heart icon for favorites, and a 'View More' button. The recipes shown are: Homemade Garlic and Basil French Fries (556.62 kcal, 45 min), The Best Chili (322.59 kcal, 130 min), Falafel Burgers (590.43 kcal, 45 min), Salmon Caesar Salad (393.21 kcal, 45 min), How to Make the Perfect Sweet Potato Sloppy Joes (679.31 kcal, 40 min), and Baked Ratatouille (1029.12 kcal, 45 min). Below the grid, there are more recipe images visible.

- Wines

Εδώ ο χρήστης μπορεί να επιλέξει αν θέλει να αναζητήσει ένα κρασί που να ταιριάζει με το φαγητό της επιλογής του ή αν θέλει να του προτείνει η εφαρμογή ένα κρασί με βάση κάποιων φίλτρων που θα ορίσει.



The screenshot shows the 'Wine Pairing' section of the SearchCuisine website. The navigation bar at the top includes the 'Foodies' logo and links for 'Recipes', 'Meal Plan', 'Wine Pairing' (which is highlighted), and 'Profile'. The main content area features a large image of wine glasses on the left. To the right, there is a section titled 'WINE PAIRING' with the text 'Find a wine that goes well with your food.' and a 'SEARCH' button. Below this, there is a 'WINE RECOMMENDATION' section with the text 'Let us recommend you the best wine possible.' and another 'SEARCH' button. The bottom of the page features a dark banner with the 'Foodies' logo and the text 'BEST ONLINE'.



- WinePairing

Εδώ ο χρήστης μπορεί να αναζητήσει ένα κρασί που να ταιριάζει με το φαγητό του, ακόμα και να του εμφανιστούν πληροφορίες για μια ποικιλία κρασιού.

The screenshot shows the 'Foodies' website's 'Wine Pairing' section. At the top, there's a navigation bar with 'Recipes', 'Meal Plan', 'Wine Pairing' (highlighted), and 'Profile'. The main heading is 'Wine Pairing', followed by a subtext: 'Wine pairing is not only about matching food and wine solely for their excellence, but also to maximize the value that each component brings to the whole experience.' Below this, there are two search boxes. The left one is labeled 'Search an ingredient, food or cuisine' and has 'steak' entered. The right one is labeled 'Search the type of wine' and has 'merlot' entered. Both have orange 'Search' buttons. In the center, there's a large image of blue grapes. To the right of the grapes, there's a text block describing Merlot: 'Merlot is a dry red wine which is smooth and medium bodied. It goes especially well with tuna, steak, burger, prime rib, and beef ribs.'

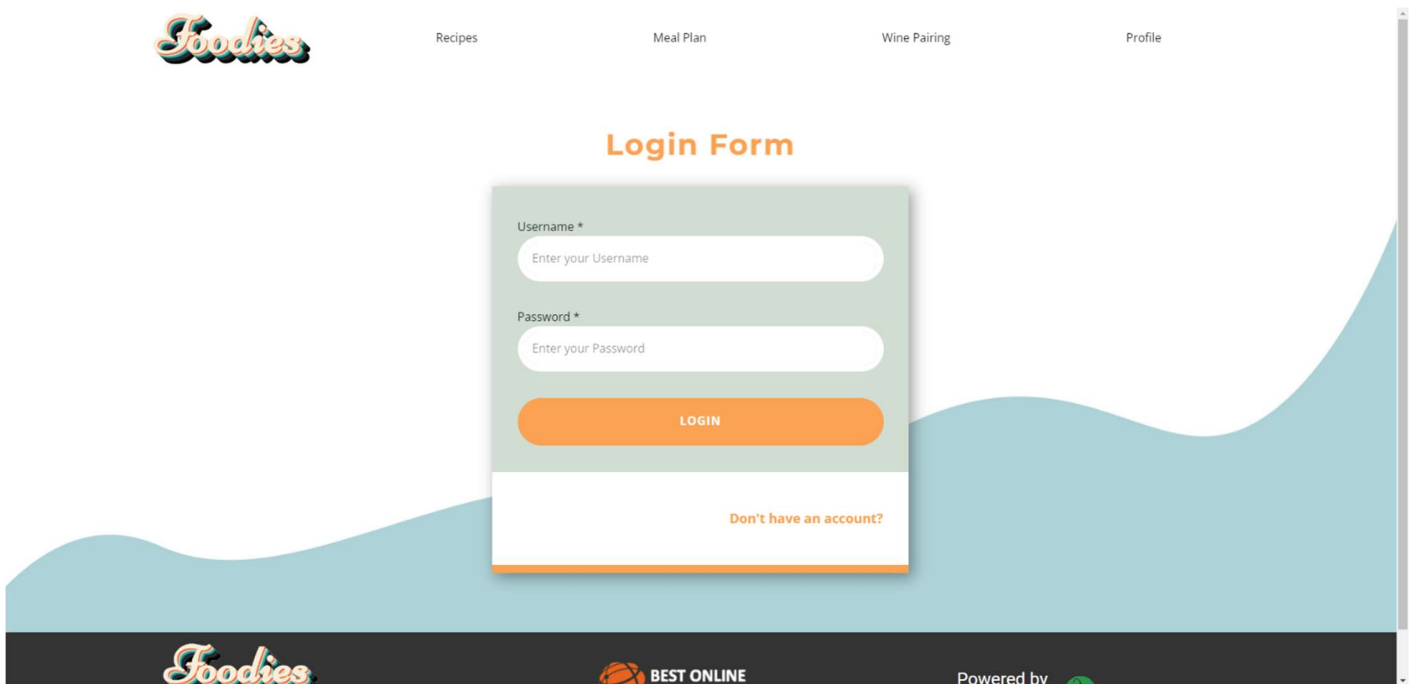
- WineRecommendation

Σε αυτή την σελίδα ο χρήστης μπορεί να εισάγει φίλτρα για να του προτείνει η εφαρμογή ποιο κρασί να αγοράσει.

The screenshot shows the 'Foodies' website's 'Wine Recommendation' section. At the top, there's a navigation bar with 'Recipes', 'Meal Plan', 'Wine Pairing', and 'Profile'. The main heading is 'Discover the best wine based on your budget and standards'. Below this, there's a search bar with 'merlot' and an orange 'Search' button. To the right of the search bar, there are two sliders: 'Max price' set to 25 and 'Min score' set to 0.8. Below the search bar, there's a large image of a wine bottle labeled 'ENATE CABERNET SAUVIGNON - MERLOT 2018 SOMONTANO'. To the right of the bottle, there's a text block describing the wine: 'Fully ruby color. A complex blackcurrant, blackberry aroma with a touch of violet and hints of red pepper, vanilla and herbs. Smooth and round mouthfeel; the soft tannic cushion gives way to a flavorsome palate which releases pleasant honey-roast aromas. The well-balanced acidity will ensure improvement in the bottle over the next five years.' Below the text, there's a star icon and the score '0.96'. At the bottom, there's a price tag '\$18.99' and a button that says 'Click here to learn more about the wine and purchase it'.

- Login

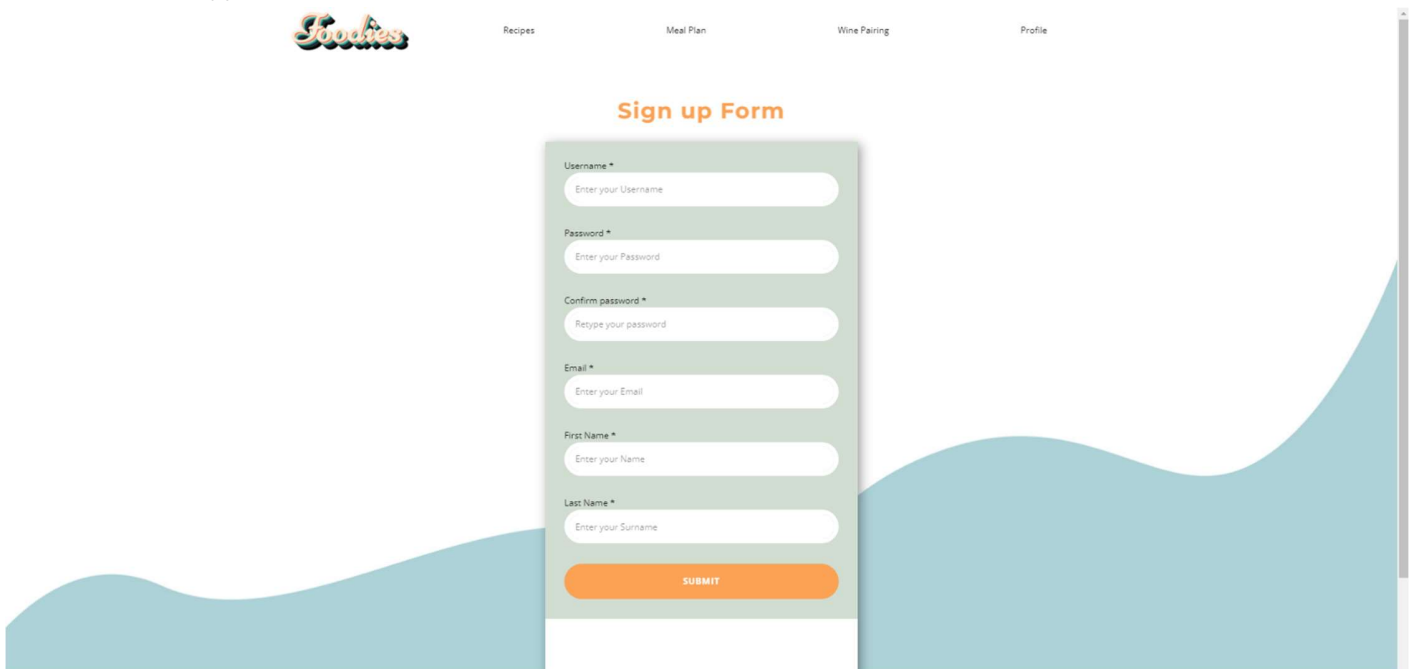
Σε αυτή την σελίδα ο χρήστης μπορεί να συνδεθεί στον λογαριασμό του με τα στοιχεία του. Αν δεν έχει λογαριασμό μπορεί να πατήσει τον σύνδεσμο “Don’t have an account” και να μεταφερθεί στην σελίδα του Sign up για να πραγματοποιήσει την εγγραφή του.



The screenshot shows the 'Foodies' website's login interface. At the top, there is a navigation bar with the 'Foodies' logo on the left and links for 'Recipes', 'Meal Plan', 'Wine Pairing', and 'Profile' on the right. The main content area features a 'Login Form' with a light green background. The form includes two input fields: 'Username \*' with the placeholder 'Enter your Username' and 'Password \*' with the placeholder 'Enter your Password'. Below these fields is an orange 'LOGIN' button. At the bottom of the form, there is a link that says 'Don't have an account?'. The website's footer contains the 'Foodies' logo, the text 'BEST ONLINE' with a basketball icon, and 'Powered by' followed by a small green logo.

- Signup

Σε αυτή την σελίδα ο χρήστης μπορεί να δημιουργήσει τον λογαριασμό του συμπληρώνοντας τα στοιχεία που του ζητούνται.



The screenshot shows the 'Foodies' website's sign-up interface. It has the same navigation bar as the login page. The 'Sign up Form' is centered on the page with a light green background. It contains several input fields: 'Username \*' (placeholder: 'Enter your Username'), 'Password \*' (placeholder: 'Enter your Password'), 'Confirm password \*' (placeholder: 'Retype your password'), 'Email \*' (placeholder: 'Enter your Email'), 'First Name \*' (placeholder: 'Enter your Name'), and 'Last Name \*' (placeholder: 'Enter your Surname'). An orange 'SUBMIT' button is located at the bottom of the form. The footer is identical to the login page, featuring the 'Foodies' logo, 'BEST ONLINE' with a basketball icon, and 'Powered by' with a small green logo.



- Profile

Αυτή η σελίδα αποτελεί το προφίλ του χρήστη. Από εδώ μπορεί να δει τα αποθηκευμένα πλάνα του και τις αγαπημένες του συνταγές και να αποσυνδεθεί από τον λογαριασμό του.

Foodies

Recipes Meal Plan Wine Pairing Profile

## Profile

HELLO, STACY

Name:  
Anastasia Mexa

E-mail:  
anastasia.mexa@yahoo.gr

LOGOUT

View your favorite recipes or your meal plans.

VIEW FAVORITE RECIPES VIEW MEAL PLANS

- MealPlan

Σε αυτή την σελίδα ο χρήστης μπορεί να δημιουργήσει και να αποθηκεύσει εβδομαδιαία ή ημερήσια πλάνα, ανάλογα με τα φίλτρα που θα ορίσει.

Foodies

Recipes Meal Plan Wine Pairing Profile

Choose the type of plan you want to create

☒ Daily ☐ Weekly

Target calories

Diet

pescetarian

Create

Total: 17.32 \$

Day	Calories	Protein	Fat	Carbohydrates
Today	1499.73	40.11	120.98	63.63

Day	ID	Title	Ready in minutes	Servings	Price	Link
Today	1697651	Caesar Dressing	5	2	2.96	<a href="https://spoonacular.com/caesar-dressing-1697651">https://spoonacular.com/caesar-dressing-1697651</a>
Today	1083981	Tuna Pasta Salad	15	8	6.00	<a href="https://homemadehooplah.com/tuna-pasta-salad/">https://homemadehooplah.com/tuna-pasta-salad/</a>
Today	542905	Chipotle Shrimp and Mango Flatbread	45	4	8.36	<a href="http://eclecticrecipes.com/chipotle-shrimp-and-mango-flatbread-giveaway">http://eclecticrecipes.com/chipotle-shrimp-and-mango-flatbread-giveaway</a>

Save





- ViewMeals  
Σε αυτή την σελίδα ο χρήστης βλέπει τα πλάνα του.

**Foodies** Recipes Meal Plan Wine Pairing Profile

Showing your meal plans

Day	Calories	Protein	Fat	Carbohydrates
Friday	1499.93	84.13	80.57	105.85
Monday	1499.93	68.02	66.65	156.81
Saturday	1499.95	83.98	51.62	172.53
Sunday	1499.96	53.81	114.57	63.57
Thursday	1500	43.31	84.21	148.42
Today	1499.92	46.82	72.48	170.7
Tuesday	1499.94	62.65	79.42	134.13
Wednesday	1499.94	65.55	107.46	67.72

Day	Title	Ready in Minutes	Servings	Price per Serving	Link
Saturday	Bacon-Cheddar Stuffed Burgers	29	4	1.97	<a href="http://www.kraftrecipes.com/recipes/bacon-cheddar-stuffed-burgers-118831.aspx">http://www.kraftrecipes.com/recipes/bacon-cheddar-stuffed-burgers-118831.aspx</a>
Friday	Jack N Jill Burgers	30	6	1.66	<a href="http://www.tasteofhome.com/Recipes/jack-n-jill-burgers">http://www.tasteofhome.com/Recipes/jack-n-jill-burgers</a>
Monday	Lemon Fresh Spaghetti with Garden Sauce & Pumpkin Flowers	30	4	4.75	<a href="https://www.foodista.com/recipe/7XZXCXG/lemon-fresh-spaghetti-with-garden-sauce-pumpkin-flowers">https://www.foodista.com/recipe/7XZXCXG/lemon-fresh-spaghetti-with-garden-sauce-pumpkin-flowers</a>
Sunday	Raspberry Jam Strips	120	60	0.10	<a href="http://www.BettyCrocker.com/recipes/raspberry-jam-strips/dfb9e6d-d251-440f-bd32-78a1355e15ec">http://www.BettyCrocker.com/recipes/raspberry-jam-strips/dfb9e6d-d251-440f-bd32-78a1355e15ec</a>
Sunday	Turkuv Tetrastini	0h	1h	1.75	<a href="https://www.theararabianbel.com/turkuv-tetrastini/">https://www.theararabianbel.com/turkuv-tetrastini/</a>

- ViewFavorites  
Σε αυτή την σελίδα ο χρήστης βλέπει τις αγαπημένες συνταγές του.

**Foodies** Recipes Meal Plan Wine Pairing Profile

Showing your favorite recipes

Title	Ready in Minutes	Servings	Price per Serving	Link
Mushroom and Vegetable Tacos	45	12	0.64	<a href="http://www.foodista.com/recipe/ZJR42WQZ/mushroom-and-vegetable-tacos">http://www.foodista.com/recipe/ZJR42WQZ/mushroom-and-vegetable-tacos</a>
Grilled Chicken Gyros With Tzatziki	75	2	6.15	<a href="http://www.foodista.com/recipe/8KQH5NQ3/grilled-chicken-gyros-with-tzatziki">http://www.foodista.com/recipe/8KQH5NQ3/grilled-chicken-gyros-with-tzatziki</a>
Pasta and Seafood	45	2	6.16	<a href="http://www.foodista.com/recipe/8YWWDKPS/pasta-and-seafood">http://www.foodista.com/recipe/8YWWDKPS/pasta-and-seafood</a>
Pasta With Tuna	45	4	1.68	<a href="http://www.foodista.com/recipe/K6QW5KQM/pasta-with-tuna">http://www.foodista.com/recipe/K6QW5KQM/pasta-with-tuna</a>
Pasta A La Lydia (Halloween Inspired)	45	10	0.55	<a href="http://www.foodista.com/recipe/35FK4GD5/pasta-a-la-lydia-halloween-inspired">http://www.foodista.com/recipe/35FK4GD5/pasta-a-la-lydia-halloween-inspired</a>
Pumpkin Almond Burger Patties	45	6	1.71	<a href="http://www.foodista.com/recipe/G6G5567G/pumpkin-almond-burger-patties">http://www.foodista.com/recipe/G6G5567G/pumpkin-almond-burger-patties</a>
Pasta With Salmon Cream Sauce	45	4	1.60	<a href="http://www.foodista.com/recipe/WRF73JT3/pasta-with-salmon-cream-sauce">http://www.foodista.com/recipe/WRF73JT3/pasta-with-salmon-cream-sauce</a>
Pasta e Fagioli (Pasta and Beans)	45	6	0.64	<a href="https://www.foodista.com/recipe/QF35KD4Y/pasta-e-fagioli-pasta-and-beans">https://www.foodista.com/recipe/QF35KD4Y/pasta-e-fagioli-pasta-and-beans</a>
Pasta With Italian Sausage	45	4	2.45	<a href="http://www.foodista.com/recipe/G8JXNCD8/pasta-with-italian-sausage">http://www.foodista.com/recipe/G8JXNCD8/pasta-with-italian-sausage</a>



## 9 Βιβλιογραφικές Πηγές

1. Όλο το υλικό των ηλεκτρονικών παραδειγμάτων και των διαλέξεων.
2. <https://spoonacular.com/food-api>
3. <https://www.iban.com/country-codes>
4. <https://www.languagetechnica.com/csharp/insert-data-html-table-database-using-jquery-ajax-asp-net-c>
5. [https://www.w3schools.com/howto/tryit.asp?filename=tryhow\\_css\\_profile\\_card](https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_profile_card)
6. <https://stackoverflow.com/questions/69841459/how-to-sum-all-the-total-in-a-table-row-js>
7. <https://stackoverflow.com/questions/5316697/jquery-return-data-after-ajax-call-success>
8. <https://api.jquery.com/jQuerygetJSON/>
9. <https://www.tutorialsteacher.com/csharp/csharp-dictionary>
10. <https://medium.com/bgl-tech/how-to-make-your-first-get-api-call-in-c-net-core-501134ee6e19>
11. <https://docs.microsoft.com/en-us/aspnet/web-api/overview/advanced/calling-a-web-api-from-a-net-client>
12. <https://lottiefiles.com/28444-hamburger>
13. <https://stackoverflow.com/questions/32643960/installing-npgsql-dll-for-postgresql>