Final report Web Technology – 2ID60 – group 7

Coen de Vente, Max van Hattum, Enzo Lucas and Thijs Visser

Hosted on: restaurantwebapp.pe.hu
Video: https://www.youtube.com/watch?v=PgsykiV9Lpc

Contents

1. Motivation

For our final project we decided to create a web app which allows users to retrieve information of restaurants, place reviews and place orders at the queried for restaurants. We also allow restaurant owners to create a page for their own restaurant and add certain information, including a menu and on online order form. This is all for free, which is in contrast to other websites where people can order food. The motivation for this project came to our group while we were exploring the different functionalities API services offer. The way we see it: A restaurant is basically a combination of different kinds of information. It has a place in the real world, it has a variety of foods to offer and it is a business. And it is a business that wants to attract customers and make profit. In the past, attracting potential costumers was done by word of mouth; visitors would tell their friends and family about their experience at the restaurant. But, since we now have the Internet, visitors can tell of their experience at the restaurant to a much bigger audience online. Sites like Tripadvisor and Yelp offer this possibility of letting people know of your experience and it were these sites we used as inspiration. However, we wanted to extend this functionality. We also allow for user to place orders directly to the particular restaurant.
Moreover, as previously mentioned we enable restaurant owners to create a page describing their restaurant and place an online menu there. This way, restaurant owners don't have to pay third parties (e.g. justeat.com and thuisbezorg.nl) a fee, anymore and the full price of their product will go to the restaurant owner.

2. Data

For our web app we use multiple API's. The most important one is the Yelp API, which is used to get the information of most of the currently existing restaurants. We use 2 datasets from Yelp, one for searching restaurants in a specific location, this is the Yelp search API:
http://www.yelp.com/developers/documentation/v2/search_api; the other one is the Yelp business API, this is used to get more elaborated information regarding a specific restaurant:
http://www.yelp.com/developers/documentation/v2/business.

To give a livelier feel to the restaurant pages, we also use the Flickr API to provide a picture related to the restaurant in question. Where possible, we want to use a picture of the actual restaurant, but if this is not possible we go down different alternatives, for example, we search for city or food.
https://api.flickr.com/services/rest/ .

We also use the Google Maps API, Google Distance Matrix API and Geocoding API, which allow us to display a map with the queried restaurant in order to give a clear view of its location. Additionally, we check the user's location with this API, which enables us to let the user decide how far away the restaurant should be from his/her current position. We also used it to place info windows at the location of the restaurant, where we can give a short overview of the restaurant in question. The Google Distance Matrix API is used to check if a restaurant orders at the postal code of a user. The restaurant can decide how many kilometers it wants to travel.
Google Maps API: https://developers.google.com/maps/web/
Geocoding API by Google:
https://developers.google.com/maps/documentation/geocoding/
Google Distance Matrix API:
https://developers.google.com/maps/documentation/distancematrix/
Furthermore, we use our own user generated data set, containing user information, reviews, restaurants, menus and orders. This dataset can be found at the same location as the rest of the application.

3. Our own API

The RestaurantApp is RESTful, and is hosted on http://restaurantwebapp.pe.hu/api. The only required parameter in every request is 'id'. If 'id' is 'all', all results will be given. This way, developers can write their own preferred search function. A parameter, which is always optional, is 'format'. This can be XML, any other value will return a JSON from.

You can get reviews, restaurants, dishes and orders.

Reviews
Returns reviews of a restaurant (or of all restaurants).

URL examples:

- http://restaurantwebapp.pe.hu/api?review=true&id=all&format=xml
- http://restaurantwebapp.pe.hu/api?review=true&id=restaurant-id

Return example (XML):

```
<results>
<reviews>
<item0>
<review>
<comment_id>1</comment_id>
<user_id>4</user_id>
<id>fictie-eten</id>
<gmt_date_time_published>2015-01-11 21:14:29</gmt_date_time_published>
<summary>Good restaurant.</summary>
<full_comment>Good restaurant.</full_comment>
<rating>4</rating>
</review>
</item0>
</reviews>
</results>
```

Restaurants
Returns restaurants. You can search by several data.

URL examples:

- http://restaurantwebapp.pe.hu/api?restaurant=true&id=all
- http://restaurantwebapp.pe.hu/api?restaurant=true&id=restaurant-id
- http://restaurantwebapp.pe.hu/api?restaurant=true&city=eindhoven
- http://restaurantwebapp.pe.hu/api?restaurant=true&postal=5037SP

Return example (XML):

```
<results>
<restaurants>
<item0>
<restaurant>
<unique_ID>15</unique_ID>
<user_id>1</user_id>
<id>eet-gezellig</id>
<phone>0135444447</phone>
```

```
<postal_code>5037SP</postal_code>
<online_orders>Y</online_orders>
<address_street>Straat</address_street>
<address_number>14</address_number>
<city>Tilburg</city>
<country_code>NL</country_code>
<name>Eet Gezellig</name>
<yelp/>
<yelp_id/>
<categories/>
<distance>0</distance>
</restaurant>
</item0>
</restaurants>
</results>
```

Dishes

Returns the dishes of a menu.

URL examples:

- http://restaurantwebapp.pe.hu/api?dishes=true&id=all
- http://restaurantwebapp.pe.hu/api?dishes=true&id=restaurant-unique-id

Return example (XML):

```
<results>
<dishes>
<item0>
<dish>
<unique_ID>34</unique_ID>
<id>15</id>
<categories>33</categories>
<dish_name>Friet schotel</dish_name>
<dish_descr>Heel veel friet... in een schotel.</dish_descr>
<dish_price>199.90</dish_price>
</dish>
</item0>
</results>
</dishes>
```

Orders

Returns amount of orders.

URL examples:

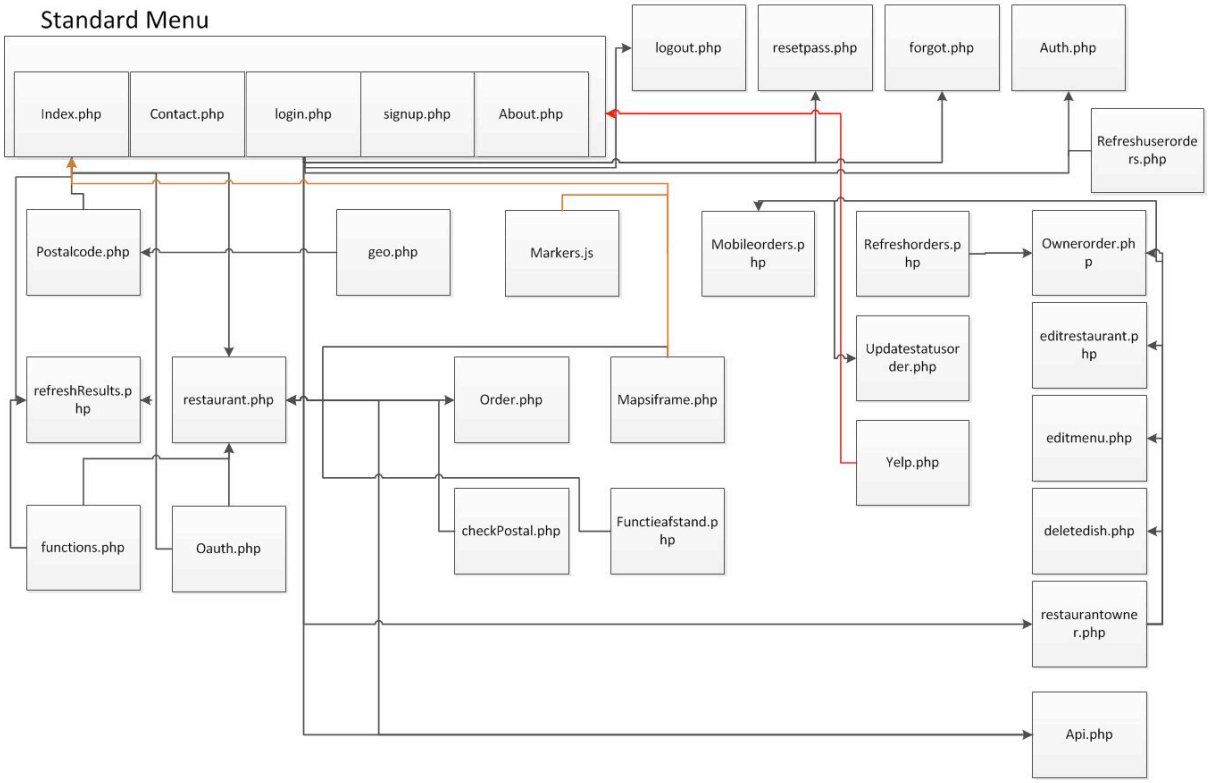
- http://restaurantwebapp.pe.hu/api?orders=true&id=all
- http://restaurantwebapp.pe.hu/api?orders=true&id=restaurant-id

Return example (XML):

```
<results>
<orders>
<orders>
<number>29</number>
</orders>
</orders>
</results>
```

4. Technical architecture

Because specific style of technical architecture was mentioned in the assignment, we have generated a table which shows how the different interfaces lock together.



The key concept here is that there is a central menu, accessible from all pages. When you are signed in, lots of extra functions will become available. When you are on specific restaurant page, you will also be made aware of this. We kept everything a bit basic with regards to infrastructure.

The framework we use is just PHP native with a database. This database communicates with our application through MySQL. It keeps track of users, with their username, email and password (also, it harbors a function to reset your password in case you lost it); information about menu categories; orders with district ids, statuses, delivery information and payment method. Also, there is a restaurant option in which you can store basic information, specific categories of food you serve and if you deliver order and how far you do this. Our database also stores the reviews not made by yelp and associates them with a user and a timestamp. Finally, there is a dishes tab which stores information about the dishes served by a restaurant such as name, price and category.
Our own API uses the information in the database in the same way as it uses the yelp information. Moreover, there is virtually no distinction between them. The only difference is that the review from yelp and the reviews made on our site are separate. Other than that there isn't much difference, only that our database has more additional functions, such as ordering as explained above.

5. User data

As mentioned a couple times before, we track the user information on several occasions. This start from the moment the user visits our site. The site lets users search by location, and is able to find the location where the user currently is. Also, we let user create an account, which offers extended functionality. A registered user can write reviews, and can become a restaurant owner and add information to his or her own restaurant page. Also, registered users are able to place orders directly to restaurants that have the option do to that.

6. Reflection

We believe this course has succeeded in transmitting its learning objectives. All of us have learned interesting new concepts regarding web development. Because for each team member their background knowledge was different, some of use learned more than others of course, but overall we would say it was a successful experience.
We do want to mention that in our opinion the lectures were not really contributing to our knowledge of web technology. It was moving too much matter in a too small amount of time and in the end, most of the information we needed we would find online anyway. Therefore, we would like to suggest that for this course a list of tutorials is posted online (on Sakai for example) instead of lectures and focus more on the workshops. We believe this would be more efficient and more enjoyable for the students.

As for the learning individually:
Coen: What I learned the most about during this course is probably APIs. My knowledge about this subject was basically zero. I learned some useful background information about APIs, the Internet and the World Wide Web during the lectures. I learned to make API calls through XML and JSON, using PHP. I gained knowledge about how APIs are usually built and how to use them properly. I had some experience with building websites already, when I stared with Web Technology. PHP, JavaScript, jQuery, AJAX, CSS and HTML are languages I had some experience with, already. However, also in the field of these languages, I gained quite some skills. Especially about nice functions, for example css(), setTimeOut() and on() in jQuery and urlencode() in PHP. I also learned MySQLi, after recommendation of a teammate. I also learned how to use GitHub, which turns out to be a very easy way to collaborate on a project with several people.

I would think that I am the person in the group who spent the most time on this project. At least, I had the largest contribution to it, according to https://github.com/WebTech7/RestaurantWebApp/graphs/contributors. The course has cost me more time than most other courses. I liked to follow this course and work on its projects.

I did encounter some problems along the way. The slow hosting of the website was one of the largest. It turned out that the main cause of this problem, was the MySQL database. First the database was hosted on db4free. Now it is hosted on the same server as the website. This improved the speed of the application quite a lot. It is still not the fastest website on the Internet, but that would be caused by a few problems that are not in our power to change. The pages that take long to load use two to three APIs and our own database at the same time. Thereafter, it combines the data from these APIs, which all adds up some processing time. We also uploaded the application to Heroku, but this was just as fast as our current host (hostinger.nl). Most other problems I can think of were coding problems and easily solved with a quick search on the Internet.

I have some ways to improve the course. I think some of the lectures, especially the ones in the beginning, did not have very much to do with the course, since we were not tested on the vast majority of these lectures. It is a good thing to give some background and side information, but this was maybe a bit too much. I can imagine that the tutorials on Friday and Tuesday were useful for people who had no knowledge about it, but when you already have a bit of knowledge about it, it is much too basic. It could be an idea to give some more advanced tutorials, as well. Also, I personally did not consider the lectures on Tuesday, the 3rd and 4th hour, where students were able to work on their assignments, useful, since it was always faster to look up the question on the Internet than waiting for the teachers

to be done with other person's questions. So might be a good idea to be there with more teachers. Despite these things, overall, I considered the course good and challenging enough.

Max: What I've learned working at this project, is to properly coordinate who codes what and to make sure everyone's code interacts in an efficient good way.
I also had to read up a lot about php, mysqli, sql, json and xml. Since it was a long time ago that I've made a website. Luckily there are enough sources on the internet that explain almost everything in an understandable way. This was particularly handy for working with web API's, since I had never done this before. Everything about php, mysqli, sql, json and xml I read about on w3schools. I find that they explain everything very clear and it is nice to see the examples and code instantly.
I also had to read up on web API's, first I read about the difference of rest and soap (http://blog.smartbear.com/apis/understanding-soap-and-rest-basics/), after this I looked through some examples that used the openweathermap API on github and with this in the back of my mind I started trying to access API's on my own.
On how to make my own API I found this tutorial very informational: http://www.brenelz.com/blog/how-to-create-a-simple-api-with-php-and-mysql/ . This gave me an idea on how to approach the idea of delivering data in jsonformat to developers.

Enzo: For me, it was very difficult to get this course started. I had little experience with web development so it was a lot to take in. the lectures were all a bit vague and not really useful in my opinion. I was able to get a better understanding of the subjects through online learning environments like codeacademy.com. In the end though I felt like I knew what I was doing, which is always a nice thing to realize. I believe I am now able to make a functional html site with database communication and use several API services.

Thijs:

During this course I learned something about the following topics:
- Basics of HTML and CSS: I didn't have any substantial prior knowledge on these two languages, so I had to learn all the basics, as well as some more advanced uses of HTML.
  In terms of CSS I only used it a little bit because I was using Bootstrap for my individual project. For both of these languages I learned the absolute basics from

codeacademy.com, and if I encoutered any problems I searched the internet for answers and found them most of the times.
- Javascript: I did have a little bit of knowledge about this before the course started, but I learned more about it and how to combine it with HTML. Also this the basic course from
codeacademy.com for this.
- PHP: I didn't know anything about PHP before, now I know a little bit how it works and I can do some basic functions with it as well as understand it. Also took most of the codeacademy.com course.
- API's: I learned what API's are and how you can implement them. I had some difficulties in understanding how they work at first but in the end it worked out well.
I used the Google Maps API and geocoding API, so now I know a lot about how those two work.
- Working on projects like these in teams: for me this was the first time working on code together with others, I learned about working in a team, giving a presentation and using Github.

Suggestions for improvements on this course:
- For me (and I've heard the same from some others), the lectures were not always very useful. Sometimes they were too difficult or went too quickly for me as a beginner to understand.
Other times the topic of the lecture didn't seem to have a direct connection to the assignments for this course. I understand that balancing this can be difficult because of the wide variety in background of the students, but maybe there should be just a bunch of online lectures that you could choose from, and the student can choose what topics he need to know more about.

7. Effort

When it comes to effort, we can honestly say we spent a lot of time on this project. Far more than the usual amount of time spent on other courses. We do have to be honest when we say that we did not contribute equally to this project, mainly because of different levels of background knowledge among team members.
The estimation of the effort of individual team members:
Coen: 40%
Max: 30%
Enzo: 20%
Thijs: 10%

However, this estimation is mostly based on results delivered, and not so much on time spent on the assignment, because for some this assignment was more challenging than for others.

Coen really was the driving force behind this project. Having had a decent amount of experience in web development, Coen was able to quickly state what was required and also able to implement this. This is why he has the largest amount of work contributed in this project. Also, Coen refers to himself as a perfectionist, which meant that there was according to him always a point of improvement within our web app.
Max was Coen his right-hand man, providing the whole team with additional support. He too had a bit of experience with web development, and was therefore easily able to fix problems. Coen and Max both contributed a lot to this project. But because the main ideas usually came from Coen his share is higher.
Enzo did not have a lot of experience with web development, but had practiced it a bit during secondary school. It was because of this that his share was not that influential, because anything he did, Coen and Max were able to do faster (and usually better). In the beginning he was really involved with the project, but his attention wavered at the end.
Thijs had a bit of a slow start with this project. He had even less experience than Enzo so it was very hard for him to catch up with the others. Because Thijs had neglected to review his mail in the beginning of this quartile, making communication difficult. When this all was sorted out however, he tried to help whenever he could.

8. Improvements after feedback

Of course we took the feedback to heart and looked for ways to improve our application. We basically improved some minor aspects, such as a better distance related search, but our main focus point was the ordering part of our own web API, since this was the main point of concern.

Our own API is an application which is able to allow users to create an account and store reviews for restaurants made by the users as previously described. The additional focus is now that when you login you can claim ownership of a restaurant. When you do this you can control a variety of factors. Of course you can alter the basic information as stated before but you can also allow user to make orders directly to you by using our site. The process works as follows: you set your restaurant to accept orders. After this, you can add dishes to your restaurant. These dishes are specified by name, category, description and price. For example, a pizza Quattro fromagi, Italian, pizza with 4 cheeses, 2.00. This places the dish on your unique restaurant page. Users then can visit your unique restaurant page and select this pizza from the menu. The user fills in his postal code and our app checks if the given postal code is not too far away. If it is not, the user needs to fill in additional information and after that the order is complete. The restaurant owner then has a special page where he can see all the orders places and the information of the customer. He then has the ability to assign statuses to the orders such as: delivered, preparing or on the way. The users can keep track of this status by clicking the icon next to his name.
It is in this way we have improved our order service.
Also a smaller update is the ability to link your restaurant on our site to your restaurant on the yelp site. This way, the Yelp restaurant will not be shown in the search results anymore and pictures from Yelp will be shown on your page.

As stated on the feedback, our web app was a bit slow. This was due to our hosting. We moved our application to Heroku but the performance didn't chance noticeably. You can verify this at: http://agile-headland-6568.herokuapp.com/postalcode.php. This link is not up to date. On http://restaurantwebapp.pe.hu is the up to date version! In order to increase performance we did chance the location of the database. This was first hosted on db4free.net, but was transferred to the same server as the rest of the application. This did improve the performance.

Also, we combined the sign-up and login page as suggested.

9. Screenshots

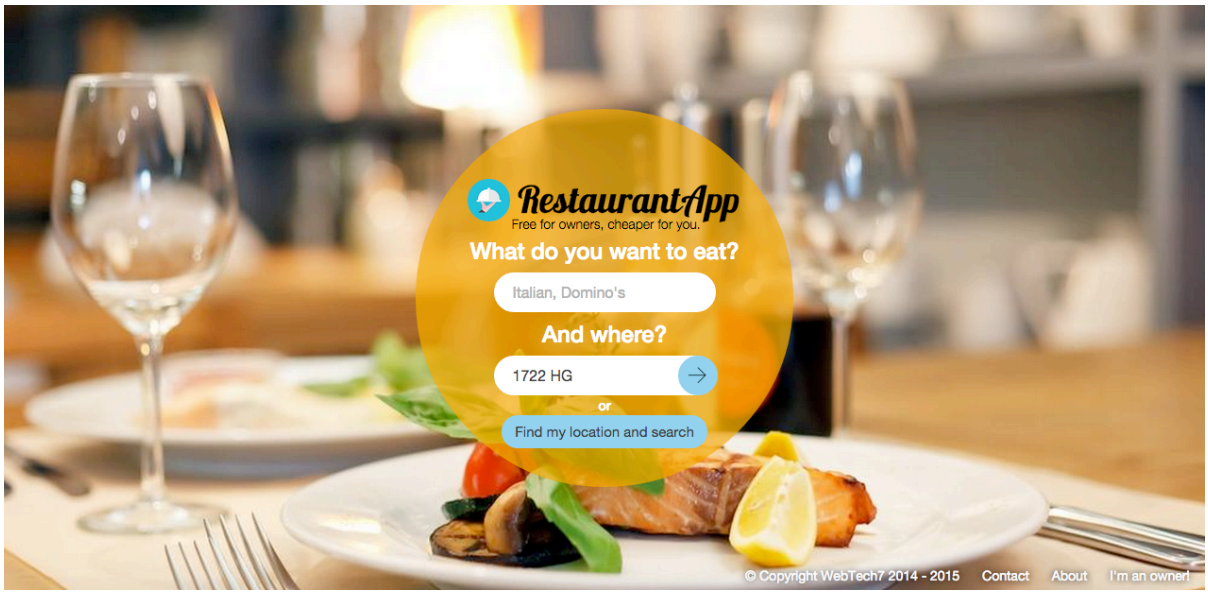


Figure 1: Front page. You can type in what you want to eat (what sort of food, what sort of restaurant or restaurant name) (optional) or where (country, postal code, city, street etc.). You can also click "Find my location and search". Accept in your browser that we are allowed to know your location and it will automatically search in your area.

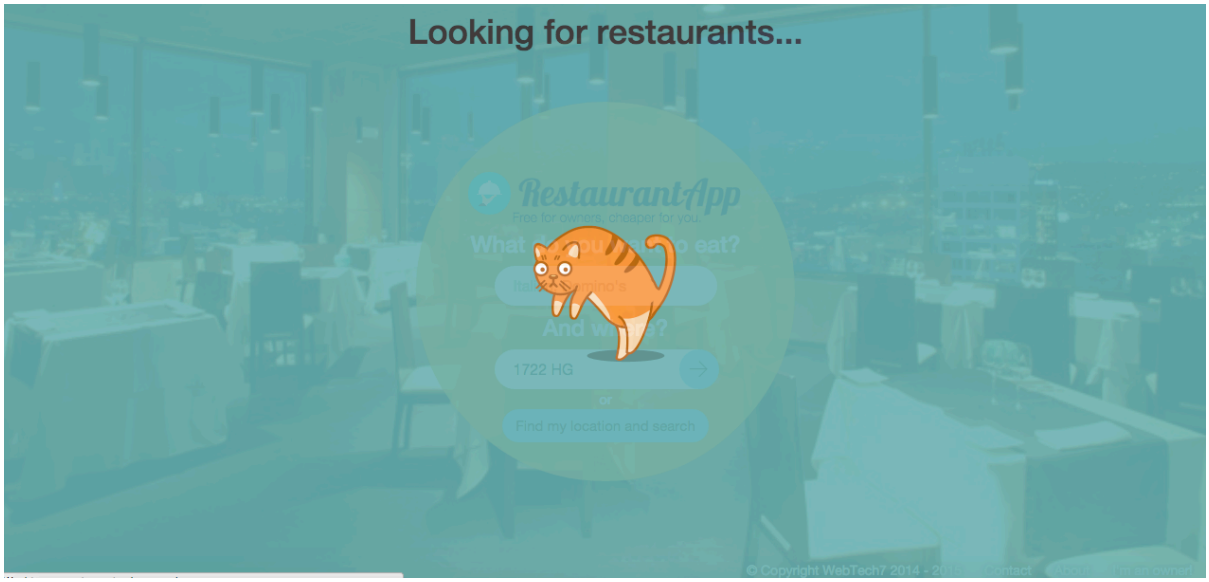


Figure 2: What you see when loading.

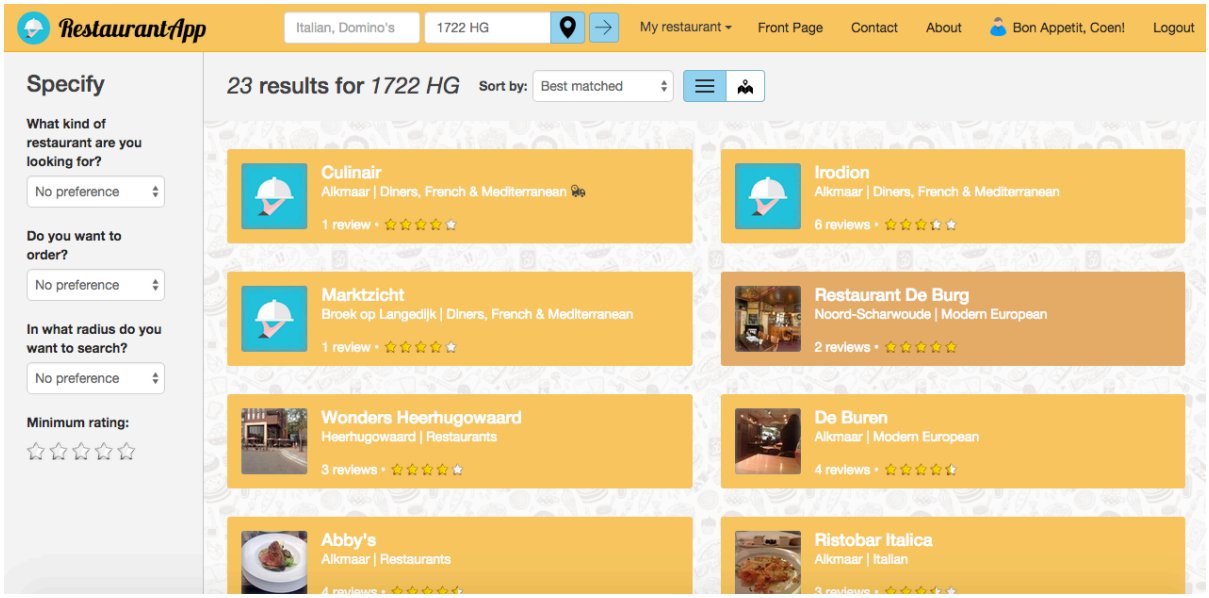


Figure 3: The page you get after the front page and when you would go to the website again (it saves your location and preference of food via cookies). In the left you can specify your choice. At the top is the navigation menu. In the middle the search results and above that, a sort option. Loading on this page is all jQuery (so no refreshing of the page).

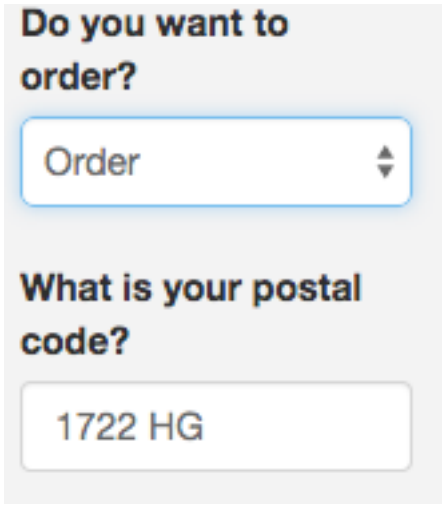


Figure 4: You can choose to order. Than you'll have to type in your postal code (if you did not search by postal code) and you will only see restaurants that order in your area.

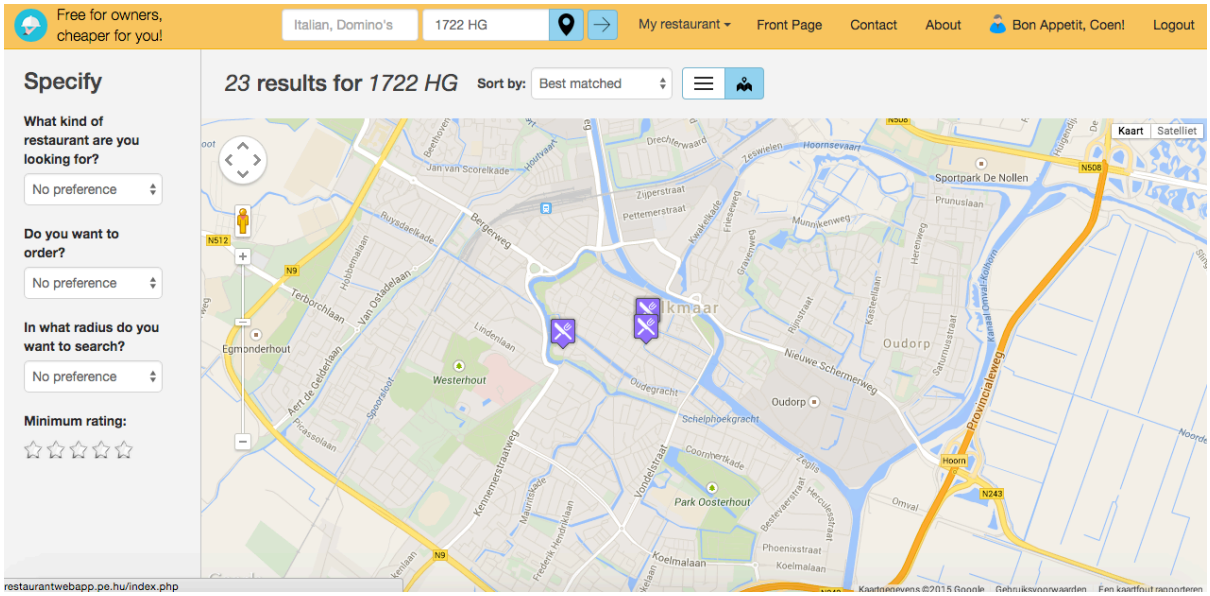


Figure 5: When you click the map icon at the top, you will see the results in a map.

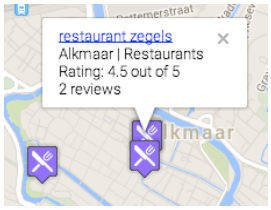


Figure 6: When you click an icon, you will get info of the restaurant.

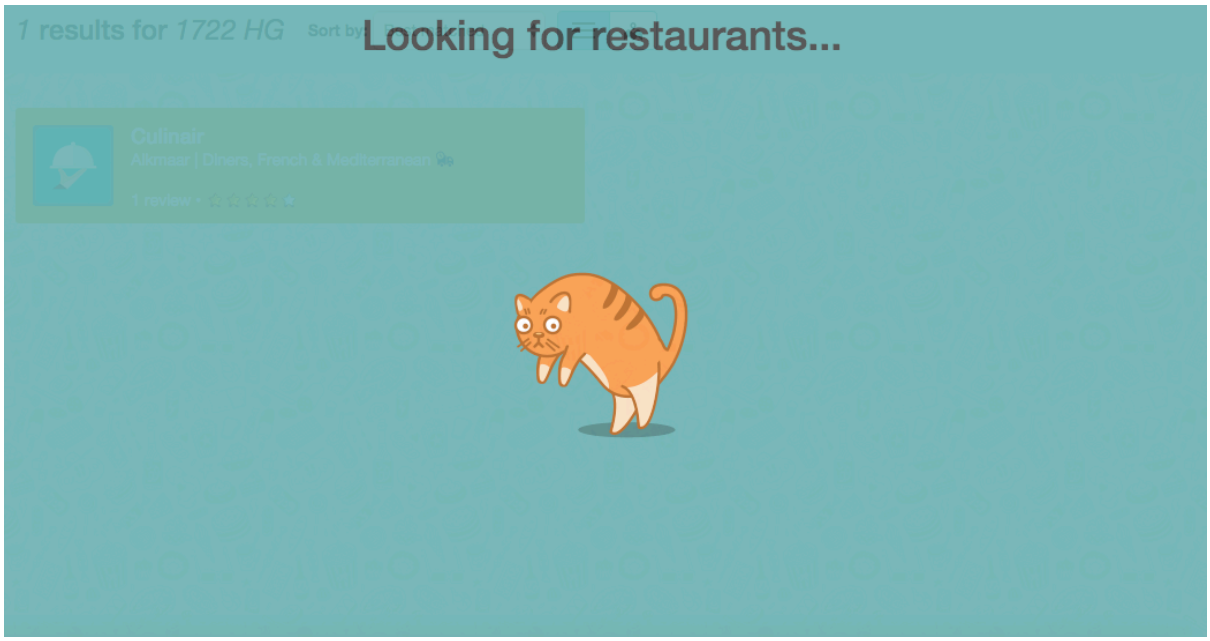


Figure 7: While loading, when you clicked a restaurant.

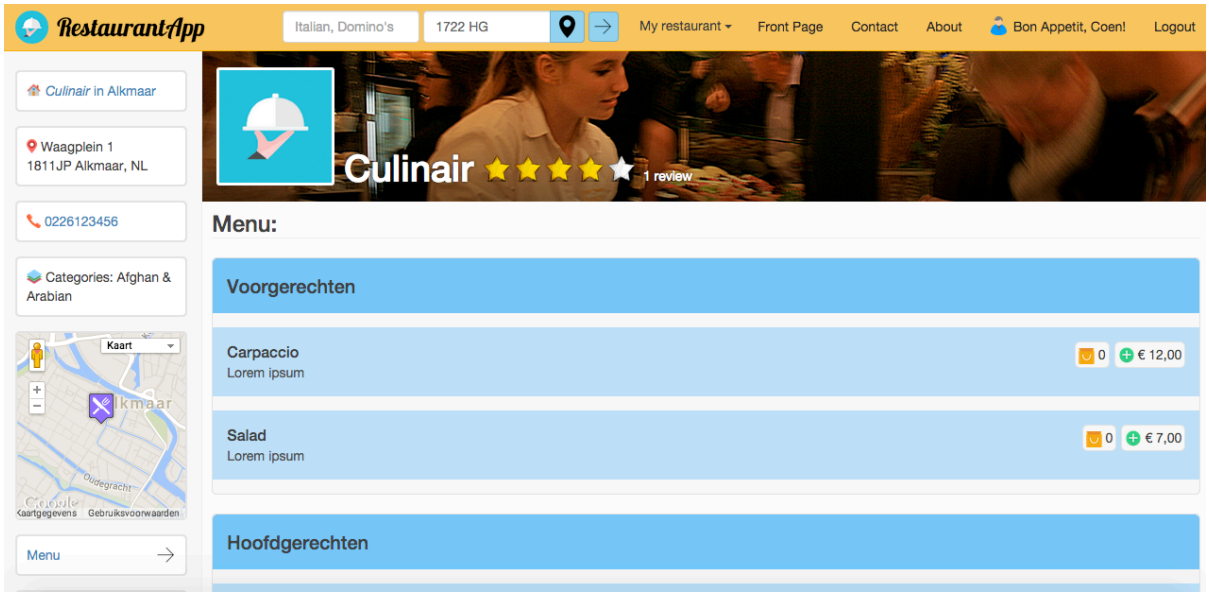


Figure 8: An example of a restaurant. On the left you see basic information about the restaurant and (if available) a link to Yelp. In the middle you will see a menu (if available). The picture at the top is from Flickr and the picture on the left is there it is available. Else it will show this default picture, you see here. The stars are calculated from Yelp and from our own reviews.

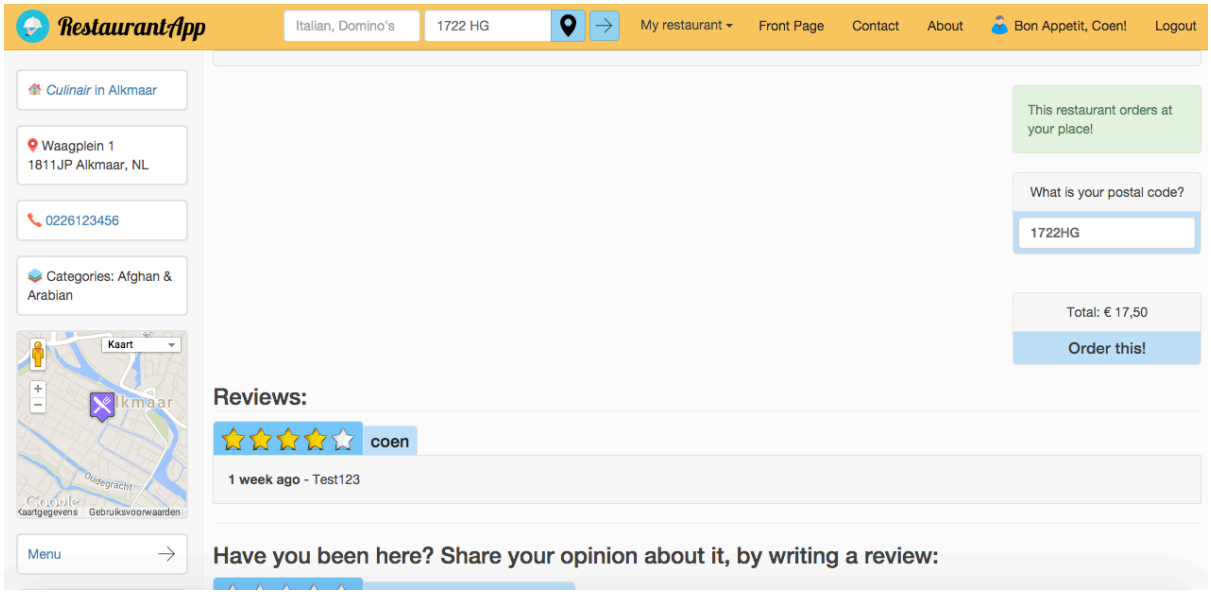


Figure 9: If you scroll down, you'll see this. On the right you see the total of your selected order. Selecting from the menu, checking the postal code and calculating the total is all done by jQuery (so no refreshing of the page). You also see the reviews. These are from Yelp (only one is from Yelp [if available], since they do not provide more) and the rest is from our own database.

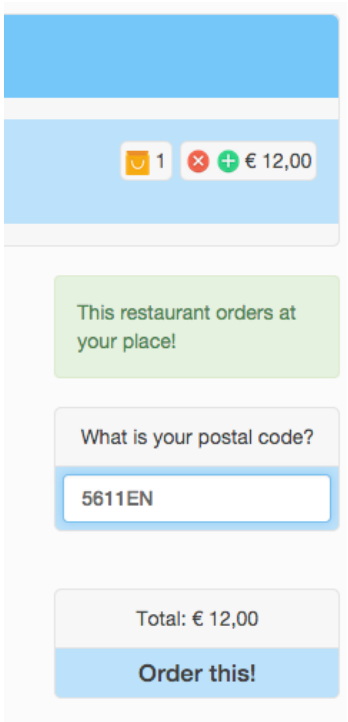


Figure 10: Here you can order.

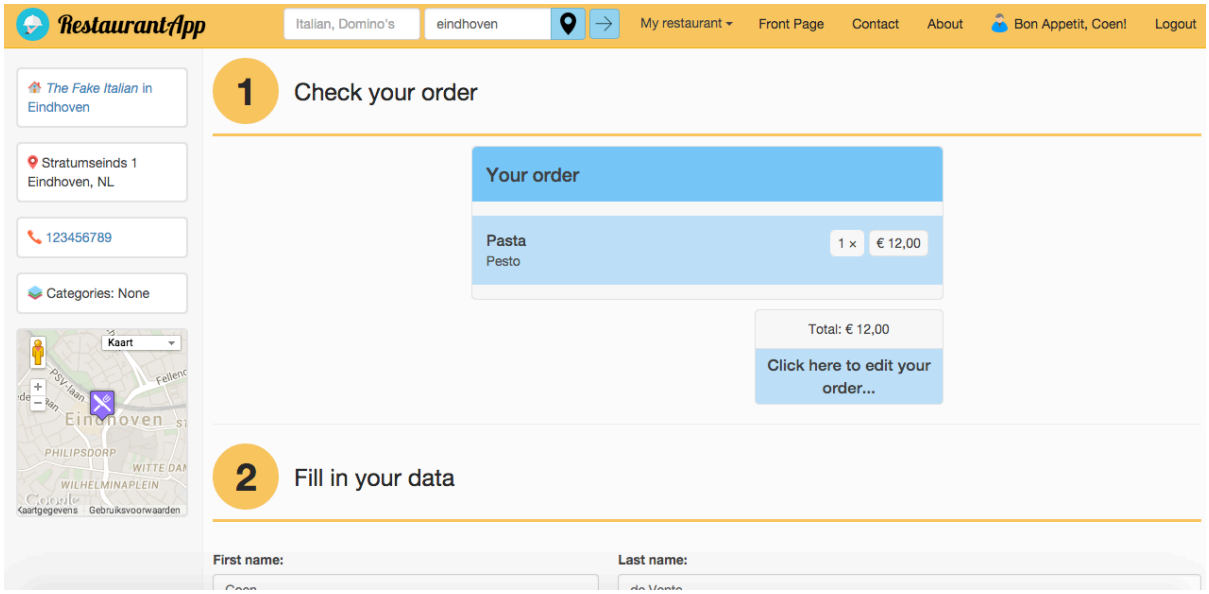


Figure 11: When you click "Order this!", you'll get here to check your order, fill in your data and submit.

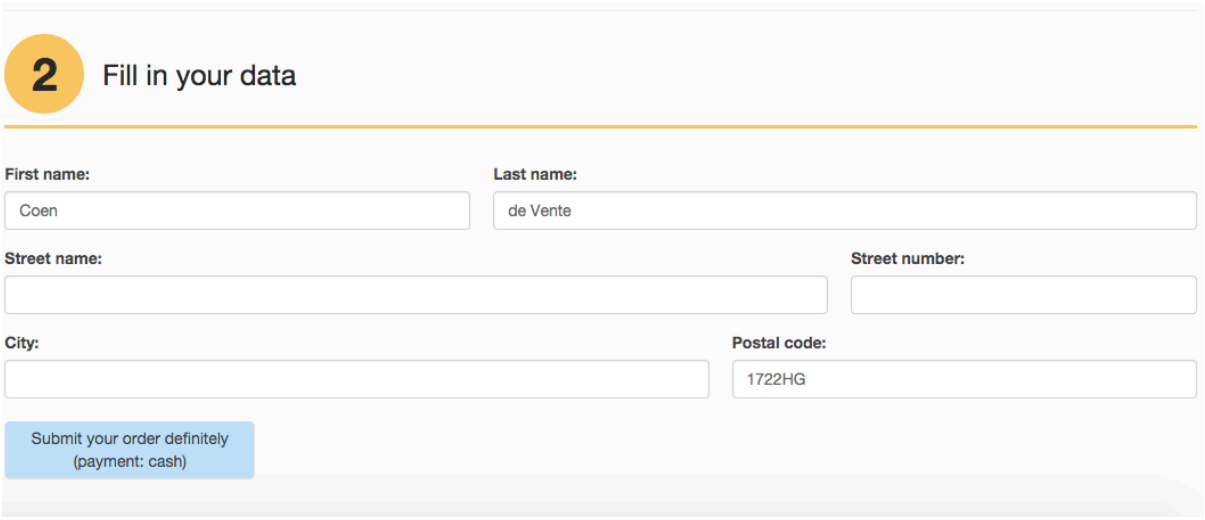


Figure 12: The fill in form looks like this.

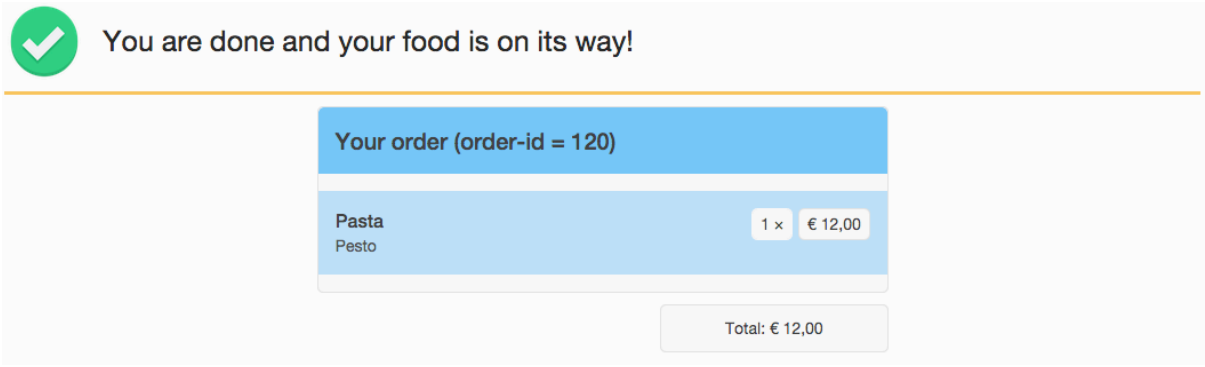


Figure 13: You will see this, when you are done (Clicked submit in the previous page).

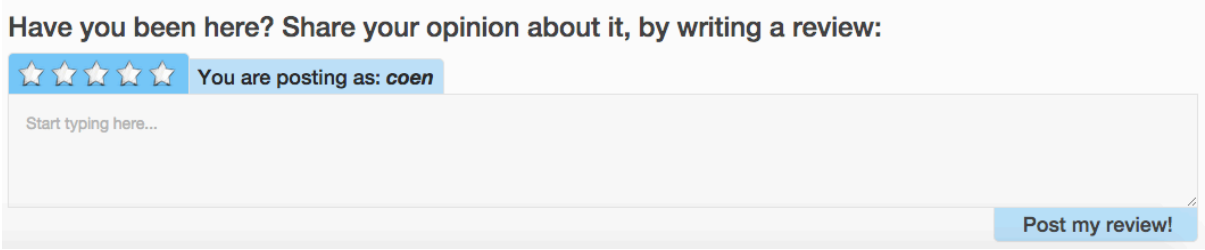


Figure 14: At the bottom, you can place your own review (if you are logged in).

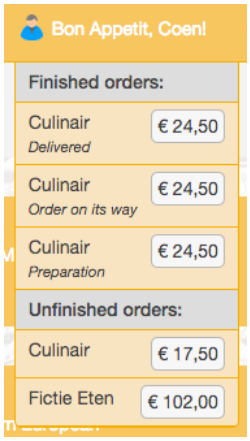


Figure 15: Finished orders can be shown when logged in. Unfinished orders can be shown always, since these are stored using cookies and finished orders are user account dependent.

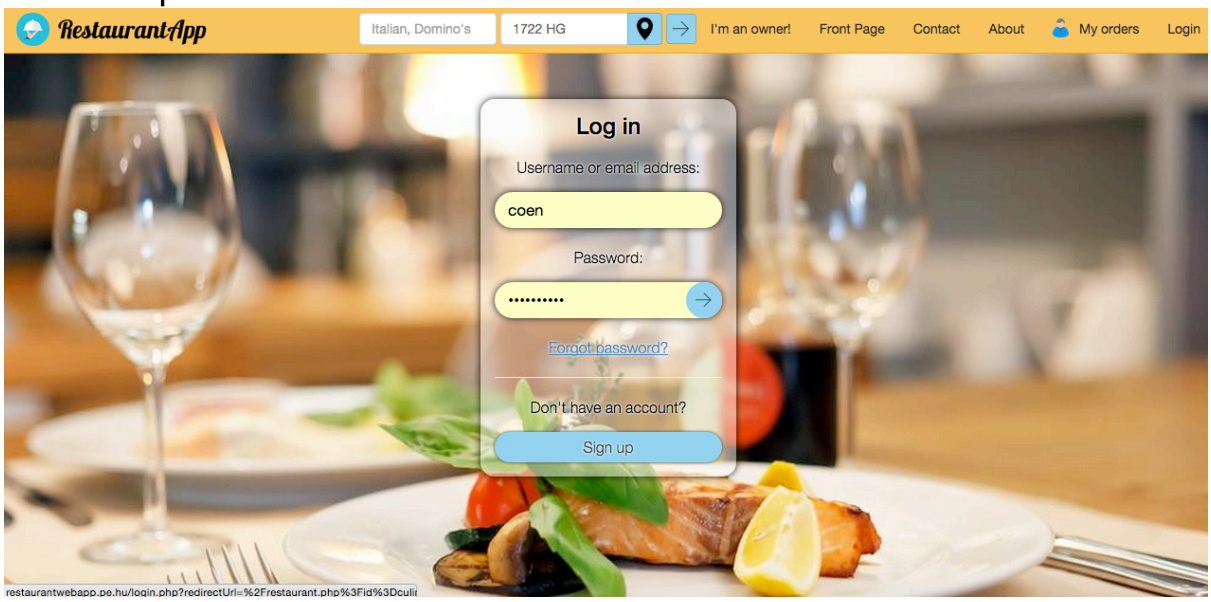


Figure 16: Via the navigation menu, you can get here (the log in page). If you do not have an account, you can click Sign Up.

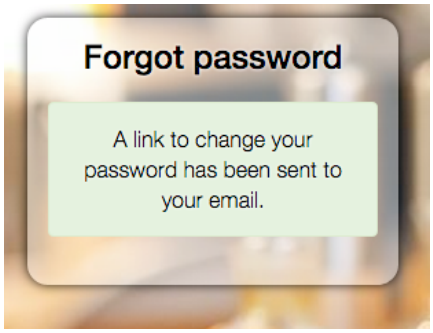


Figure 17: You can also click "Forgot password" in the login form to send an email to change your password.

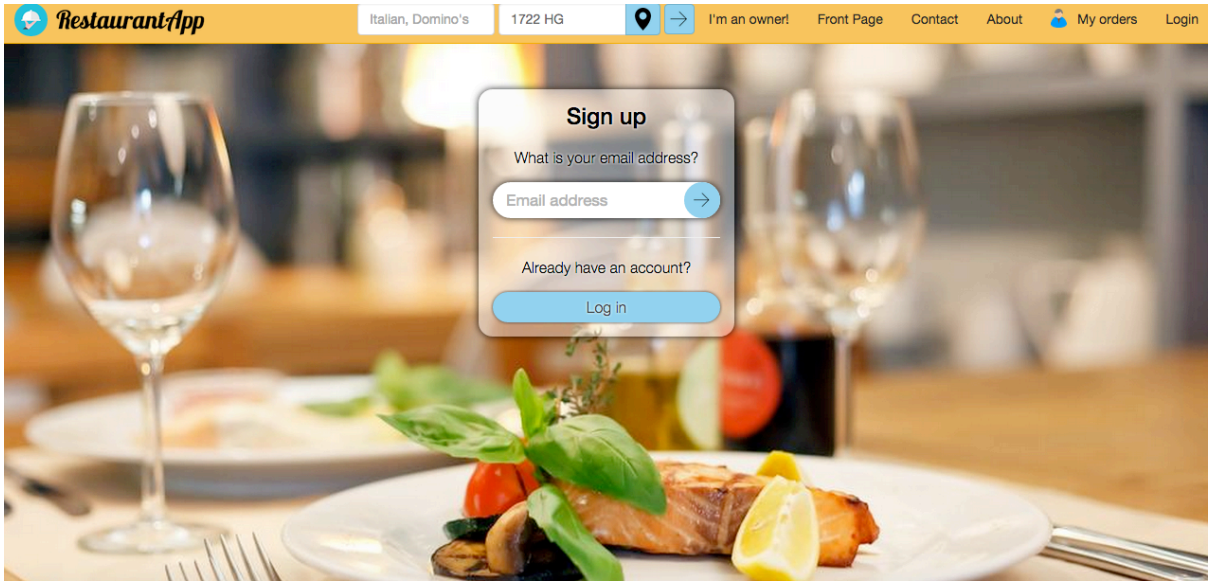


Figure 18: The sign up page. It will send a verification email. After this email you will fill in all your information. You can also resend your email if the email already exists, but is not verified.

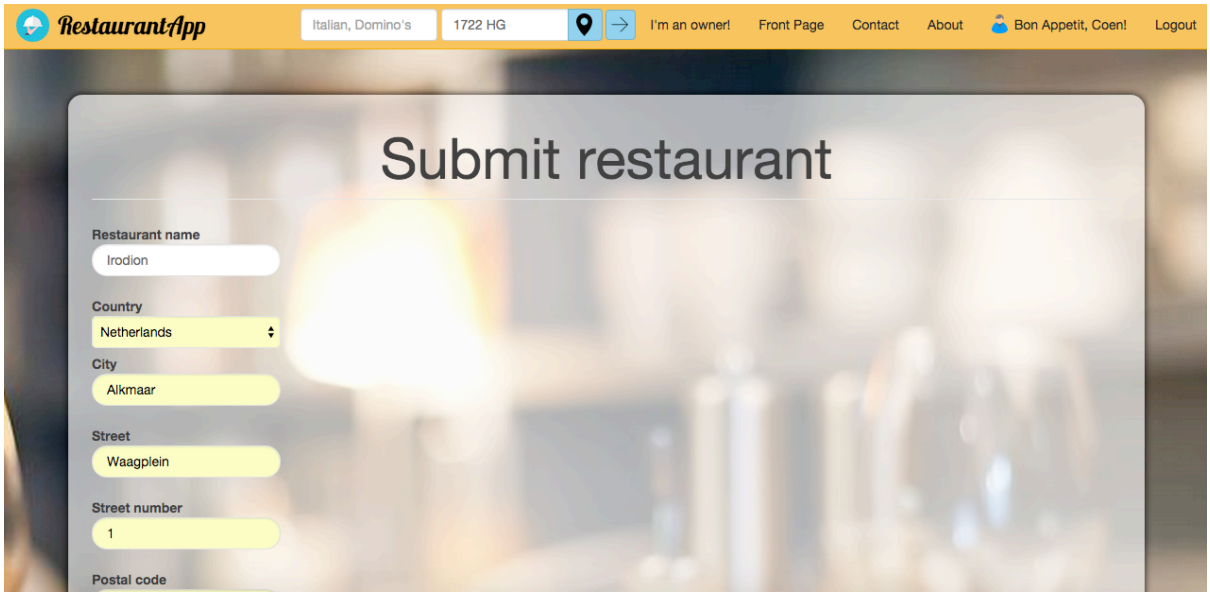


Figure 19: If you are logged in, you can add your own restaurant. If you already have a restaurant, you can get to a similar page to edit your information.

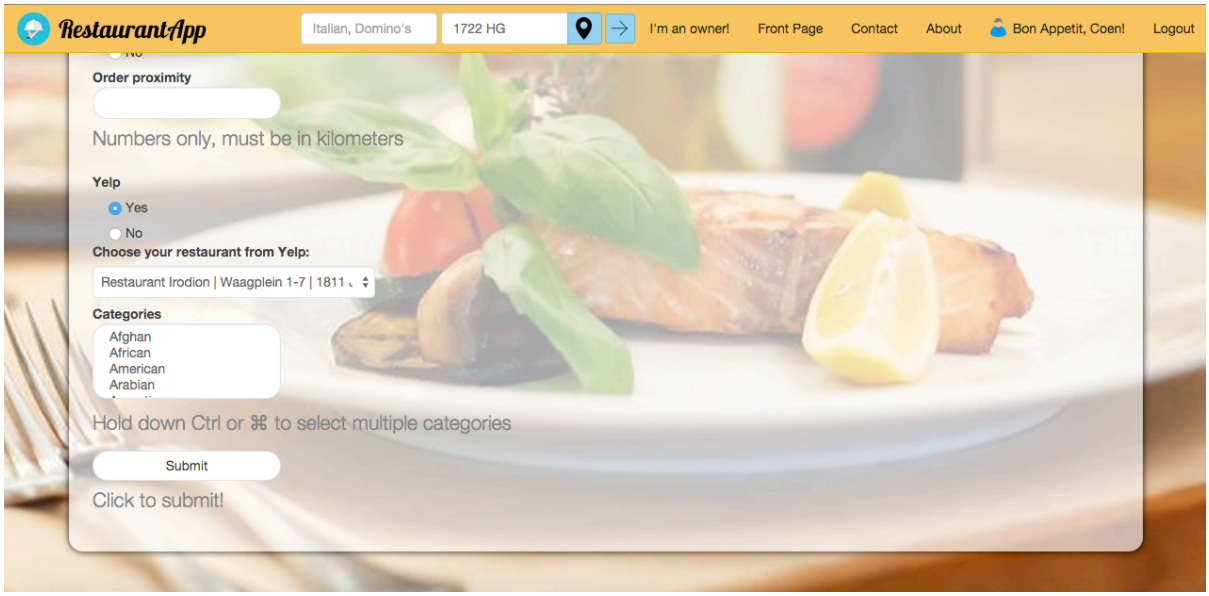


Figure 20: You can link to yelp (done via jQuery and AJAX and the Yelp API) while submitting or editing your restaurant.

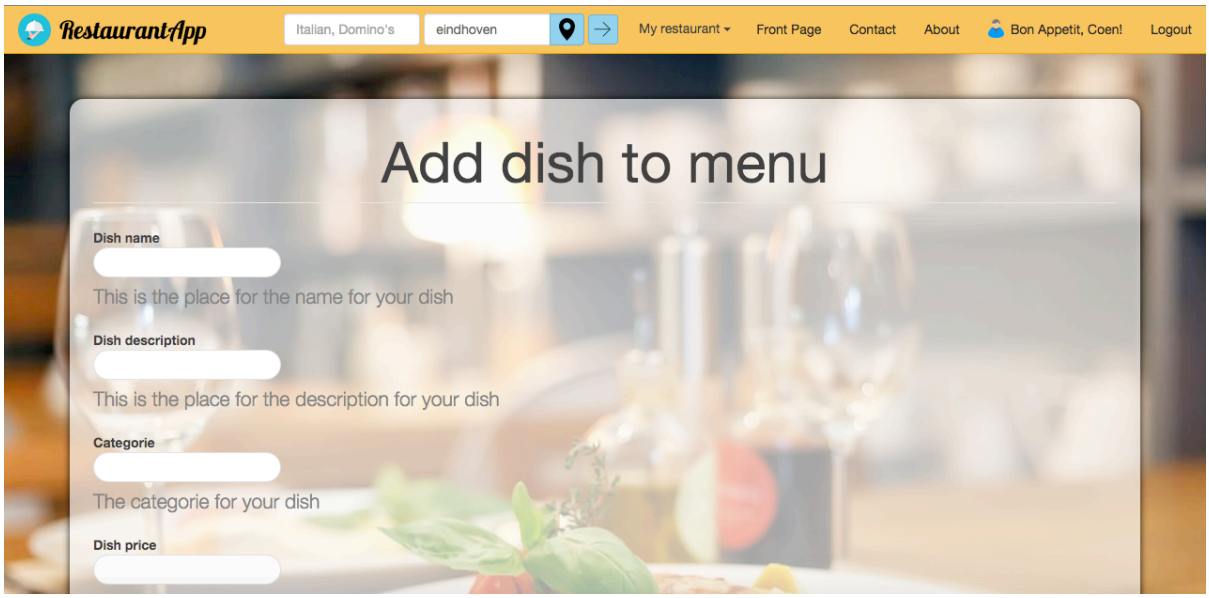


Figure 21: You can edit your menu, by adding dishes...

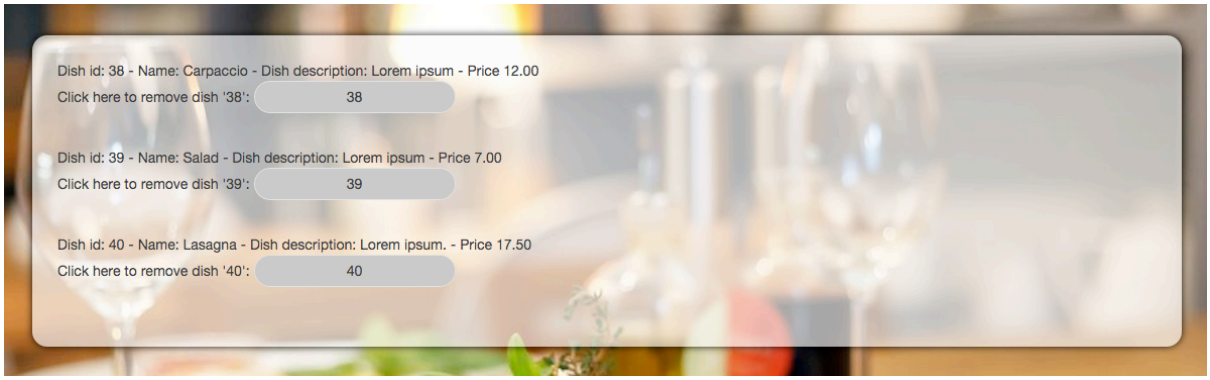


Figure 22: ... and deleting them.

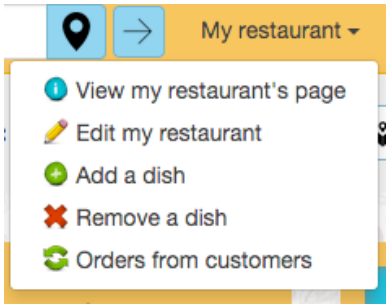


Figure 23: This is the menu for restaurant owners.

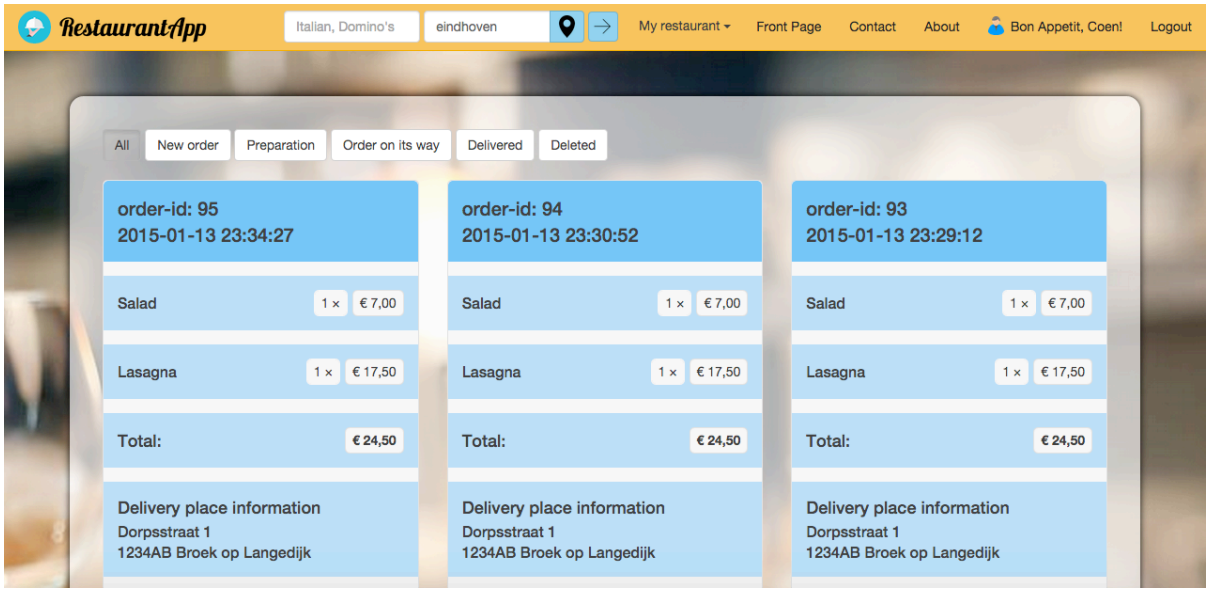


Figure 24: The page the restaurant owner has open on his iPad or computer when he is making dishes. The page will refresh automatically and constantly (via jQuery and AJAX). The owner can change the status of the orders (which the user is able to see).

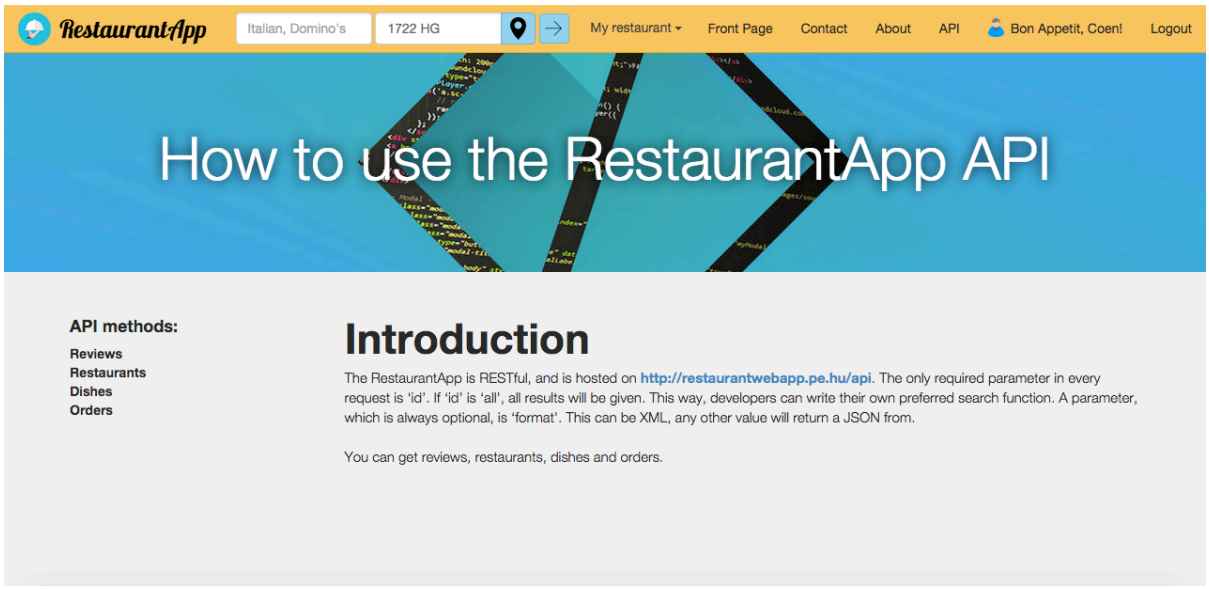


Figure 25: API manual (accessible form navigation menu).

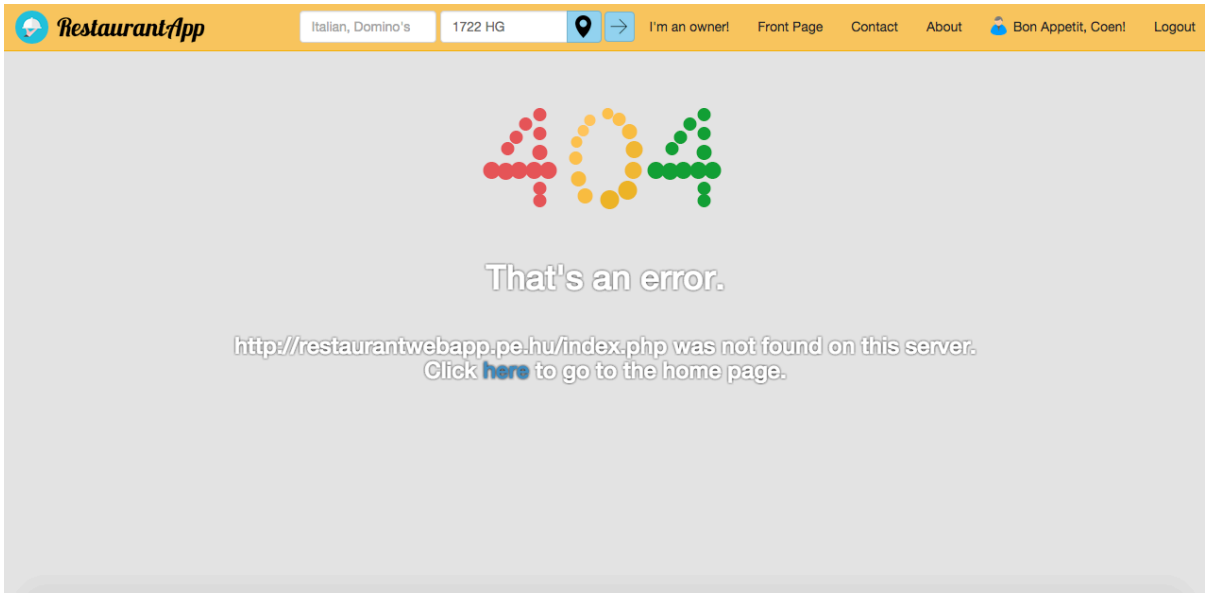


Figure 26: The 404 page. The balls in the numbers 404 are bobbly. Done by a JavaScript function from the Internet, based on a Google Doodle.