

Skill

Wappler integration

Aydan Allahverdiyeva



Introduction	3
Concept	3
Target	3
Advantages of Skill	4
Roles and Features	4
Basic user features	4
Admin features	4
Database/UML	5
User	5
Offer	6
Category	6
Subcategory	6
Comment	6
Design	7
Pages	7
Login	7
Register	8
Main	9
Home	10
UserPage	11
UserOffers	12
AdminRequests	14
Comments	16
Wappler Integration	17
Frameworks	17
Frontend	17
Backend	17
Database	18
Example of backend - frontend communication	19
Users and permissions	19
Fetch data	21
Filtering menu	25
Future development of Skill	29

Introduction

Concept

Skill is a web application, where users can acquire new skills in an exchange of their own knowledge. Users post Offers based on “I take - I give” principle. To make web application more user friendly and filtering system easier, each Offer is tied with the “I give” Category, Subcategory and Level, as well as the ones for “I take”. Figure 1 shows the use case of Skill.

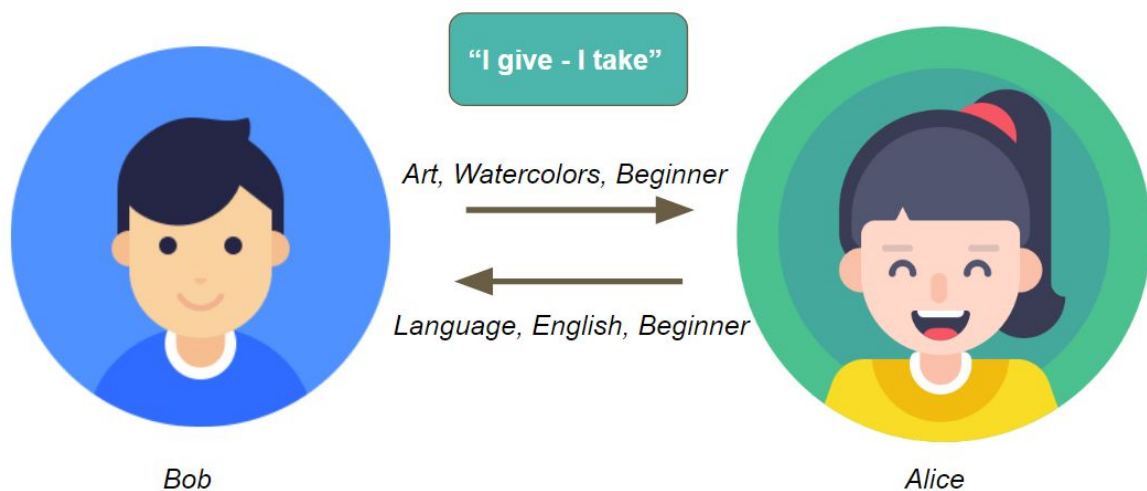


Figure 1. Skill use case. Bob placed an Offer, where he offers other users (for simplicity it is assumed that Alice responded to the Offer) watercolors lessons for beginners in exchange that Alice teaches him English at beginner level.

Target

Skill suits anyone who is willing to share the knowledge in exchange of learning new and anyone who wants to know something new but does not have a lot of money to invest. The web application may be popular among people, who do not have a lot of budget to spend yet want/need to boost their skills in various fields.

Advantages of Skill

- Meeting new people
 - Since Skill can be used worldwide, users can find new friends from all around the world
- No financial burden
 - No payment required. The only value that matters is knowledge
- Advanced search
 - Due to established categories, subcategories and levels, the filtering on the offers is highly effective

Full code is available here: <https://github.com/aydanaeva/Skill>

Roles and Features

To maintain Skill's safe and friendly environment, it is needed to decide which Offers should be or should not be posted. That is why the web application maintains two roles: administrator (admin) and basic user. The admin is also a basic user, however, the role gives more control of the web application.

Basic user features

- Post/delete offers (deletion is possible only before the offer has been posted)
- Search/filter on offers
- Comment on offers

Admin features

- approve/disapprove/provide feedback for users' offers
- add/delete categories and subcategories

Database/UML

To get a better understanding of the logistics of Skill I decided to create a UML diagram. UML diagram helped me to understand the classes needed for my database, as well as relationships between them. Please refer to Figure 2.

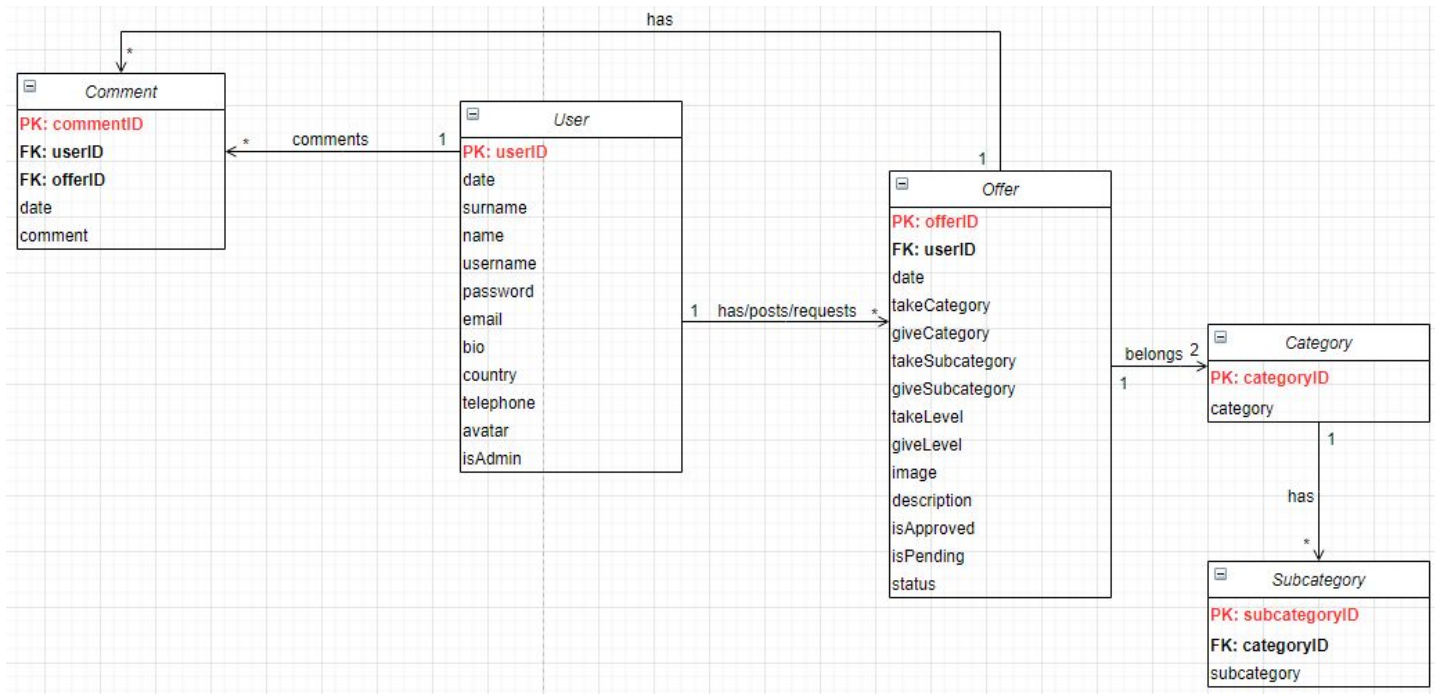


Figure 2. UML diagram that describes classes and their relations to each other in the database. (PK: Personal Key - unique personal ID for each object of the class; FK: Foreign Key - reference to other class).

User

The User Table is the most important class in the logistics of Skill, because every other class directly or indirectly depends on the userID. One user may write zero or more Comments, as well as post zero or more Offers.

It is possible to consider the admin role of the application as an aggregation class of user role. Indeed, in the initial design of the database Admin and User were two separate tables. Later, a much simpler solution was found: it was decided to maintain boolean **isAdmin**. If the user is Admin (if **isAdmin** is true), then the user has access to extra features and admin page.

Offer

Each Offer has its creator: the user. That is why the Offer Table references the userID. Besides the Offer stores “give/take” categories, subcategories and levels. Since users may post or delete the offer, and the admin may approve or reject the offer request, there are four possible scenarios. In order to differentiate between those scenarios, the table supports two booleans: isApproved and isPending. Please refer to Figure 3.

isApproved	isPending	Scenario
+	+	The offer is approved and waits to be posted by user
+	-	The offer is approved and has already been posted by user
-	+	The offer is sent to the admin for approval, thus in pending state
-	-	The offer is rejected by admin, thus not possible to post it

Figure 3. Offer's scenarios described by booleans.

Category

The Category Table maintains the list of all categories existing in the database. Each Category has a unique ID and one or more Subcategories.

Subcategory

To main parent-children relation of Category and Subcategory classes, each Subcategory stores reference to the Category it belongs to (categoryID).

Comment

Besides maintaining a unique ID for each Comment, it is necessary to reference the creator of the comment (userID) and location of the comment (offerID).

Design

I have established several requirements for the design of Skill:

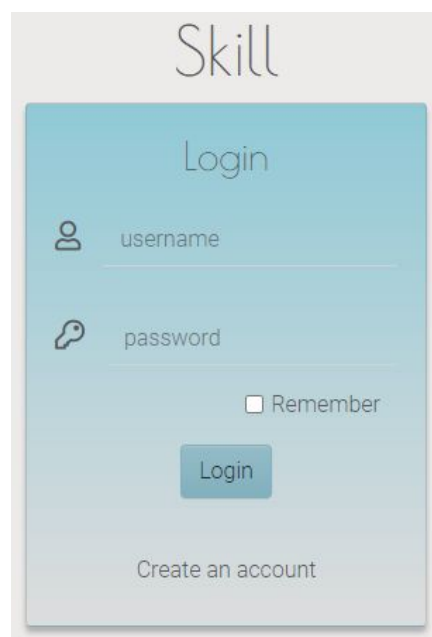
- Responsiveness of the pages
- User-friendly and pleasant interface
- Self-explanatory structure
- Support on small devices

Pages

The design scheme is subtle and smooth: rounded angles, light shadowing, soft color transitions and light text font. The main color of the application is blue and gray color on the background is complementary.

Login

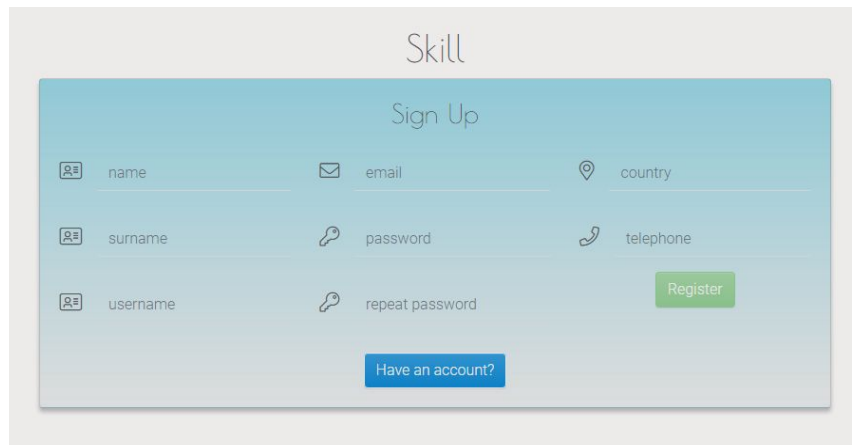
User's login page is the starting point of the application. Both Admin and Basic User use the same login form. Each user should have a unique username regardless of Caps Lock. User can easily redirect to the Register page by clicking on the "Create an account" button.



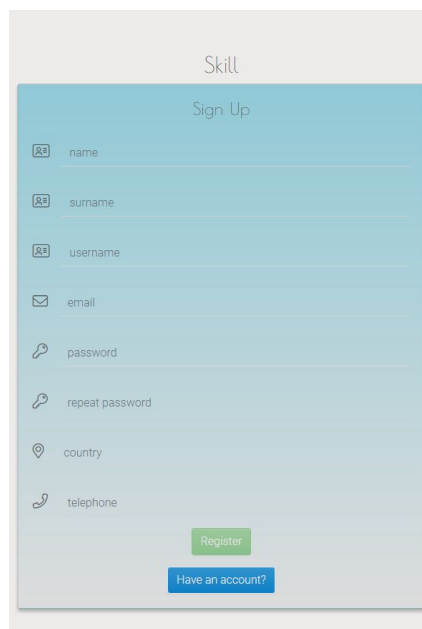
Picture 1. Login page.

Register

Even though Registration page follows the same structure and design as Login, there is an additional challenge: the form on the registration page needs to be responsive. The goal is to make sure that three columns on large screen form can easily combine to one column on smaller devices. Users are automatically logged in after registration. Please refer to Pictures 2a and 2b.

The image shows a web browser window with the title "Skill". Inside the browser, there is a "Sign Up" form. The form is organized into three columns. The first column contains three text input fields labeled "name", "surname", and "username", each preceded by a small icon of a person. The second column contains three text input fields labeled "email", "password", and "repeat password", each preceded by a small icon representing the field type (envelope for email, key for password). The third column contains a text input field labeled "country" preceded by a location pin icon, and a green "Register" button. Below the "Repeat password" field, there is a blue button labeled "Have an account?".

Picture 2a. Register page on large devices.

The image shows the same "Sign Up" form as in Picture 2a, but it is displayed on a mobile device. The form is now in a single-column layout. The input fields are stacked vertically in the following order: "name", "surname", "username", "email", "password", "repeat password", "country", and "telephone" (which was not visible in the desktop version). The "Register" button and the "Have an account?" link are at the bottom of the form.

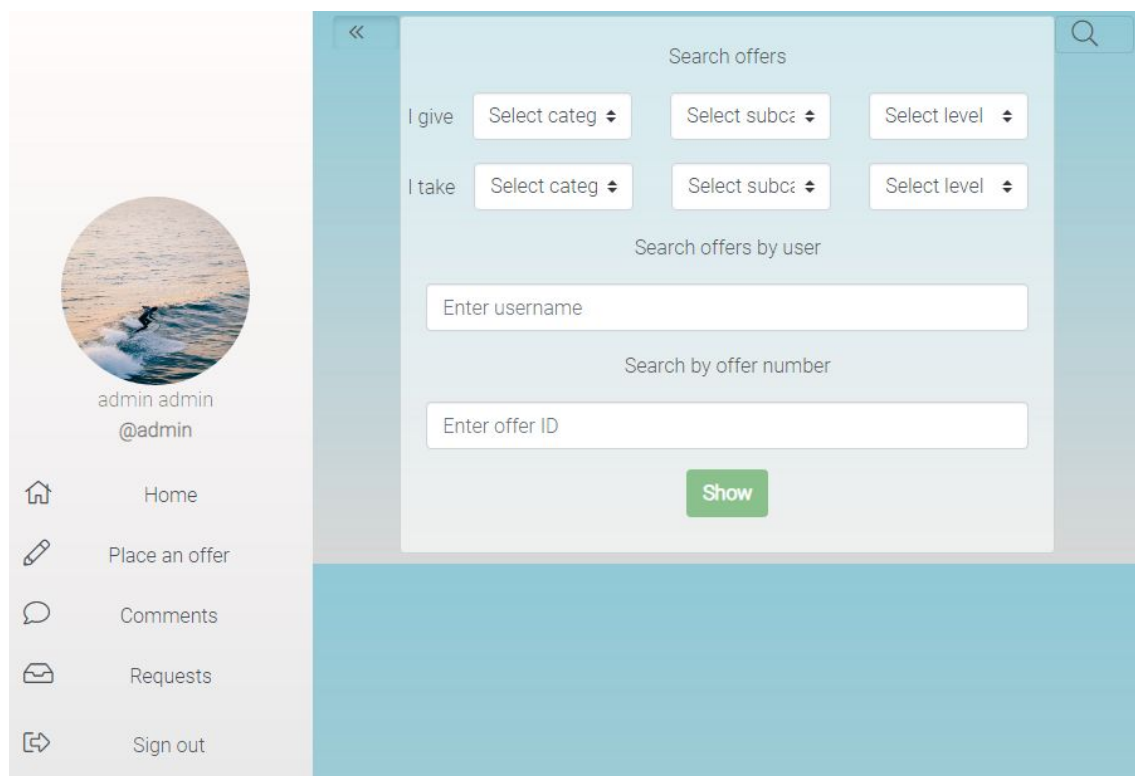
Picture 2b. Register page on small devices.

Main

Main page is layout for all pages that the user will interact with after login. The layout consists of two blocks: sidebar block and navigation bar block. Please refer to Picture 3.

The sidebar block consists of links to different pages. If the user is not Admin, the “Requests” link is hidden and access to the page is restricted.

The navigation block includes the filtering menu. Here, it is possible to filter based on “I give/I take” categories, subcategories and levels. It is also an option to get the offers posted by a particular user by entering the username. In case the user wants to search for an exact offer, the user can enter the offer's ID. The “Show” button submits the query parameters for search and toggles Dynamic Modal with results. This allows the user to search for offers on any page the user is currently on.

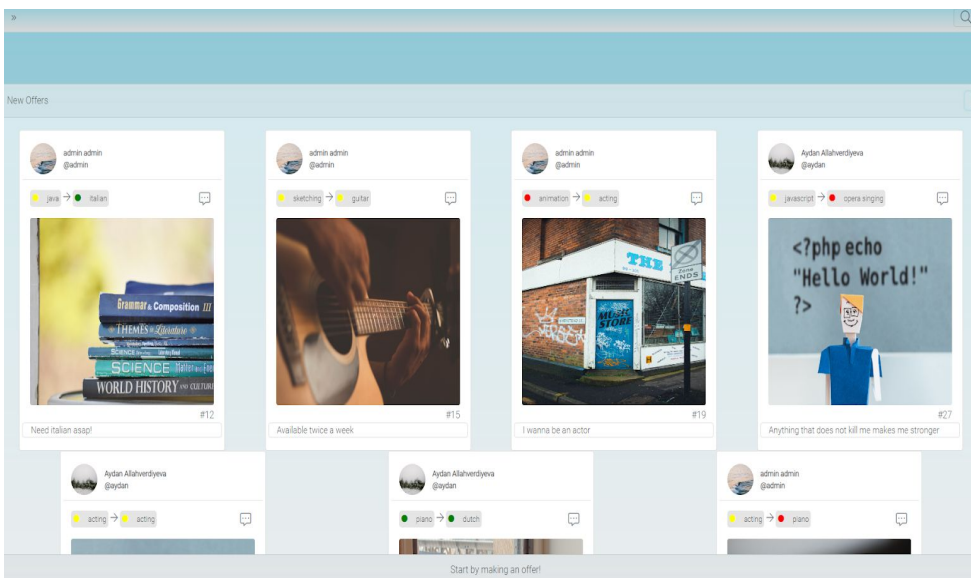


Picture 3. Main layout.

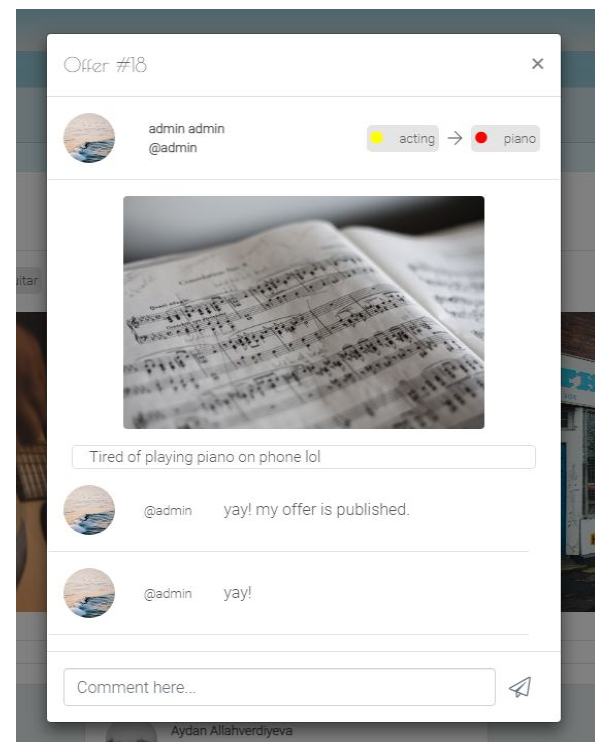
Home

After successful login, the user is redirected to the Home page. Home page consists of one card with repeating offers published by all users. With a small button in the right corner of the header it is possible to sort offers by date published (ascending, descending). Please refer to Picture 4a.

To comment on the offer or read other comments, the user needs to open a full offer. For that offer card has a “comments” icon, which toggles the full offer Dynamic Modal. Please refer to Picture 4b.



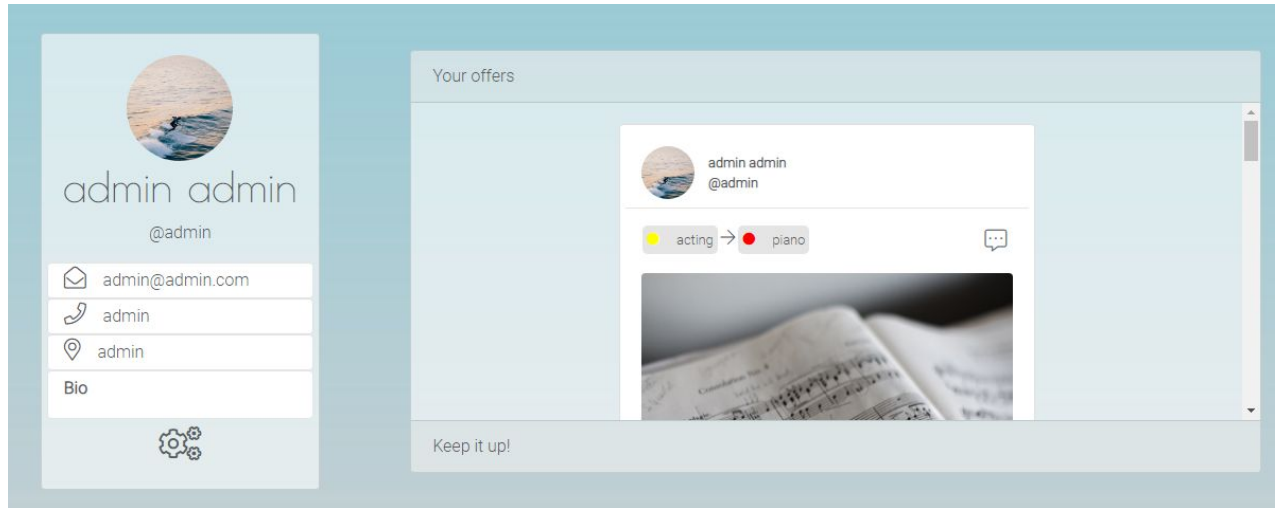
Picture 4a. Home page.



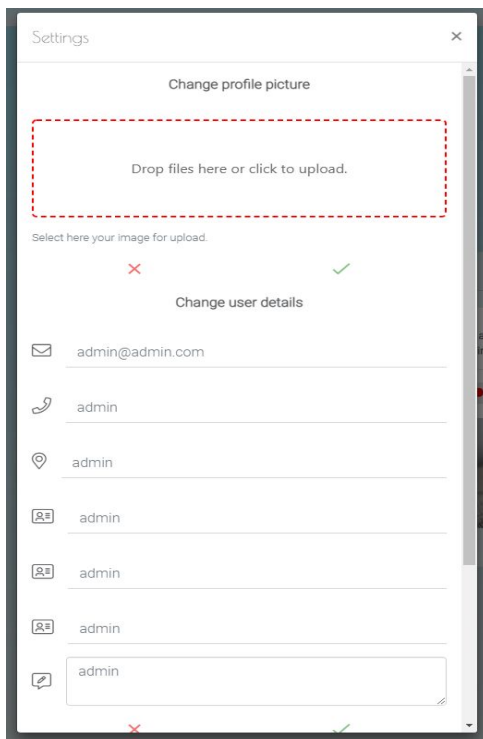
Picture 4b. Full offer modal.

UserPage

By clicking on the avatar on the sidebar, the user is transferred to their own user page, which consists of two cards: left one with user information and right one with offers that the user has posted. Please refer to Picture 5a.



Picture 5a. User page.



Picture 5b. Settings modal.

The “settings” icon on the personal information card opens a modal with user settings, where users can change their profile picture, personal information (name, email etc.) and also reset the password. Please refer to Picture 5b.

UserOffers

UserOffers is the page where users can send requests for their new offer, as well as browse their pending offers (offers that wait for admin approval, offers that are rejected and offers that are approved but not posted yet by the user). The page consists of two cards: the upper one is the form to create an offer, the lower one is repeating references to pending offers. Please refer to Picture 6a.

The screenshot displays the 'UserOffers' page interface. At the top, there is a navigation bar with a double arrow icon on the left and a search icon on the right. The main content area is divided into two primary sections. The upper section, titled 'New Offer', contains a progress indicator with two green dots and a right arrow. Below this, there is a red dashed rectangular box with the text 'Drop photos here or click to upload'. To the right of the upload area, there are two columns of form elements. The left column is headed 'I give' and the right column is headed 'I take', with a right arrow between them. Each column contains three dropdown menus labeled 'Select category', 'Select subcategory', and 'Select level'. Below these columns is a large white text input field with a red 'x' icon on the left and a green checkmark icon on the right. The lower section of the page is titled 'Your pending offers' and contains a table with three columns: 'Post: #25', 'Status:', and an eye icon.

Picture 6a. UserOffers page.

In order to create a new offer and send the request to the admin, the user needs to fill in the following information about the offer:

- "I give" category, subcategory and level
- "I take" category, subcategory and level
- Description (optional)
- Image (optional)

To simplify the design of the offer card, the levels correspond to the colors:

- Beginner - red
- Intermediate - yellow
- Expert - green

Right after the user fills in all the necessary information, the offer is sent to admin for approval. Please refer to the Picture 6b.

localhost:8100 says
Your offer has been successfully sent to our admin for approval!

OK

New Offer

vue.js → english

Drop photos here or click to upload (florian-olivo-Ek9Znm8lQ1U-unsplash.jpg)

florian-olivo-...
1.5MB

Self-taught programmer willing to teach some front-end in exchange of english classes

I give → I take

programming ✓
vue.js ✓
Intermediate ✓

language ✓
english ✓
Beginner ✓

Self-taught programmer willing to teach some front-end in exchange of english classes ✓

Picture 6b. Filled in offer request.

AdminRequests

AdminRequests page is available only for Admin and restricted to all Basic Users. The page consists of two cards: the upper one is for incoming requests from users and the lower one is for adding/removing categories/subcategories for the offers. Please refer to Picture 7a.

The card with categories/subcategories modifications menu consists of four forms:

- Form to add category (category is ensured to be unique regardless Caps Lock)
- Form to add subcategory (subcategory is ensured to be unique regardless Caps Lock)
- Form to remove category (automatically removes all subcategories related)
- Form to remove subcategory

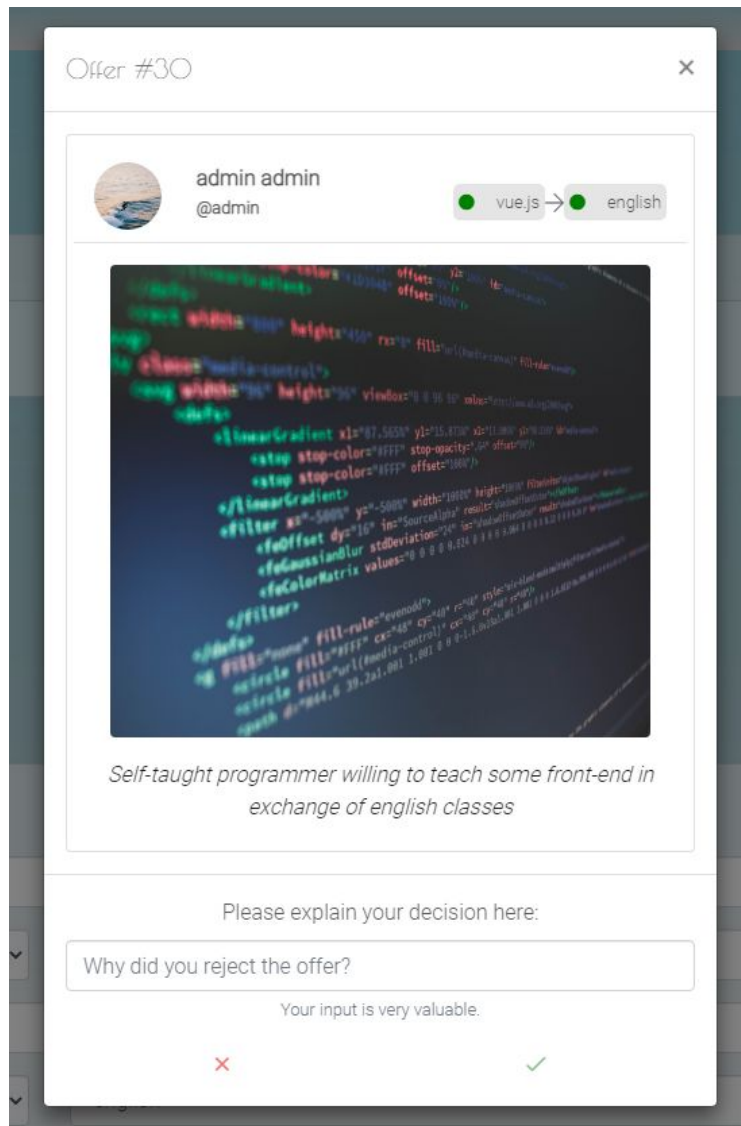
The screenshot displays the AdminRequests page interface. At the top, there is a header bar with a search icon. Below it, a section titled 'Requests' shows 'Post: #30' with an edit icon. The main content area is titled 'Modifications' and contains four forms for managing categories and subcategories. Each form has a text input field, a dropdown menu, and a corresponding 'Add' or 'Remove' button. The first form is for adding a category, the second for adding a subcategory, the third for removing a category, and the fourth for removing a subcategory. The buttons are green for 'Add' and red for 'Remove'.

Modifications		
<input type="text"/>		<button>Add</button>
<input type="text" value="science"/>	<input type="text" value="chemistry"/>	<button>Add</button>
<input type="text" value="language"/>		<button>Remove</button>
<input type="text" value="language"/>	<input type="text" value="english"/>	<button>Remove</button>

Picture 7a. AdminRequests page.

The card with offers to review has a button that toggles a Dynamic Modal to view a full offer and also send users some feedback. Please refer to Picture 7b.

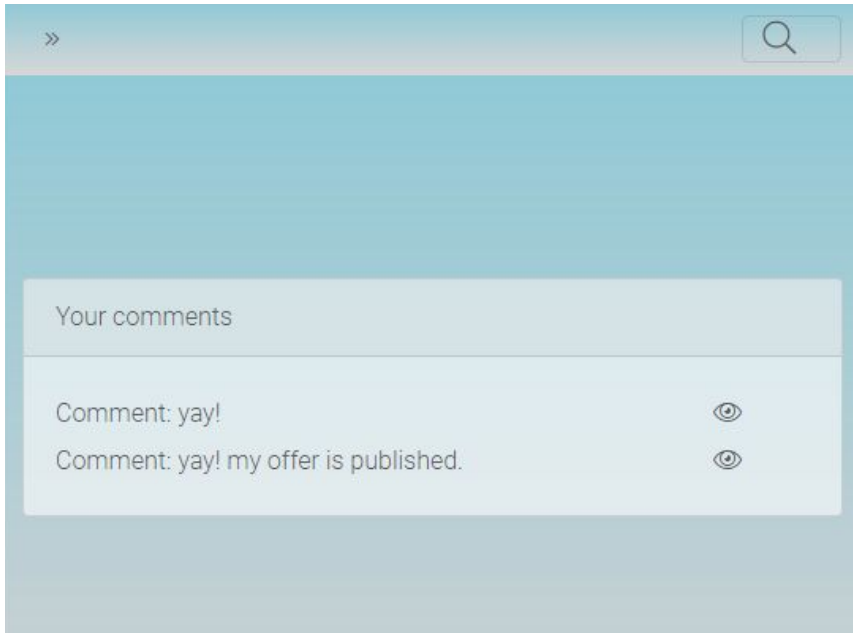
The footer of the Model gives the option for the admin to send some feedback to the user. The feedback will be sent both in case the offer is approved and rejected.



Picture 7b. Full offer review mode.

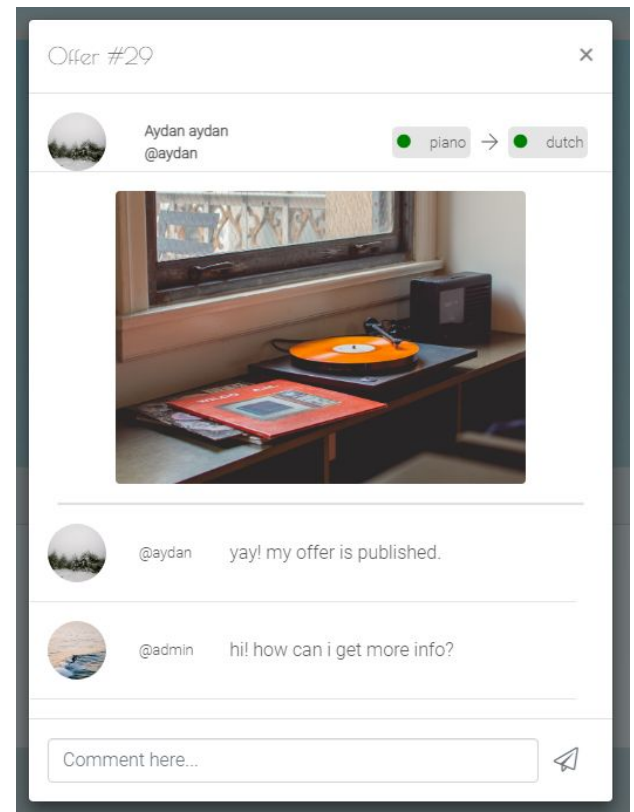
Comments

Comments page gives the list of comments the user has written. The “eye” button toggles the Modal for the full offer view mode. Please see Pictures 8a and 8b.



Picture 8a. Comments page.

Picture 8b. Full offer view mode.



Wappler Integration

Wappler supports various manager interfaces and frameworks, which I would like to discuss in this section.

Frameworks

- MySQL - database
- jQuery - frontend and backend communication
- Node.js - server actions
- Docker - container to run Node.js and MySQL servers
- Bootstrap - web design

Frontend

- Site manager
 - Provides easy access to the pages that are already separated in three folders: pages, layouts and partials
 - It is also possible to view files and folders of the project directory inside of Wappler

Backend

- Routing manager
 - Self-explanatory design
 - Allows to setup redirects and URLs for the web application
- Server connect manager
 - Current version of Wappler enables the establishment of global variables. In my project I use global security provider, database connection and identity field (for logged in user)
 - I especially enjoyed working with the security provider, since it is possible to put restrictions on pages, as well as add simple login and sign out features.

-
- I also used the Scheduler option in my project to schedule automatic updates for offers, categories etc.

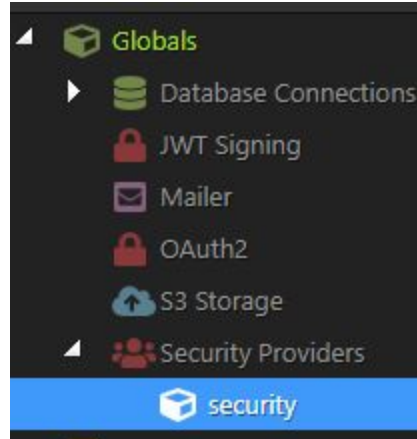
Database

- Database manager
 - Self-explanatory design
 - Ability to setup database, as well as seed it
 - Queries/inserts/updates/deletion of the data is handled on server connect manager

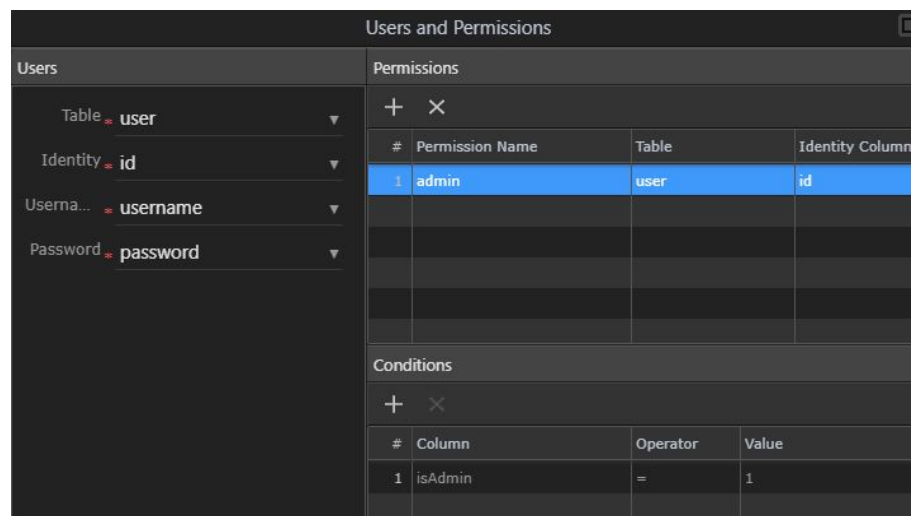
Example of backend - frontend communication

Users and permissions

1. In order to restrict admin functionalities, as well as the admin page for basic users, I have established "admin" user permission (if a user's isAdmin field == 1 -> access approved) in the security provider. Please refer to Pictures 9a and 9b.

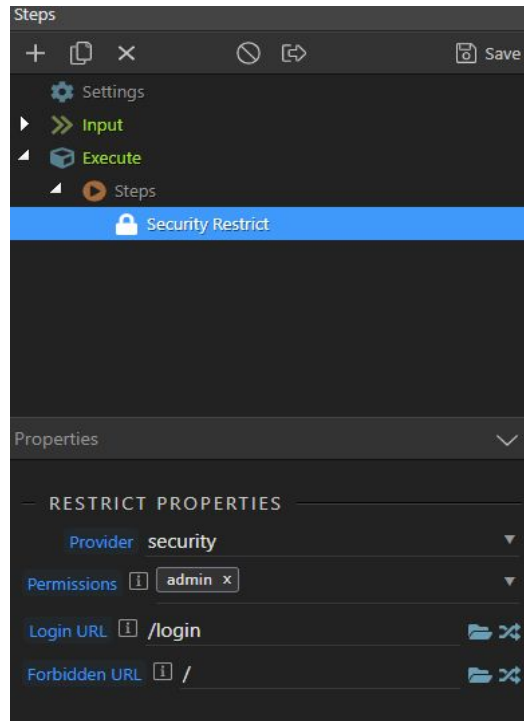


Picture 9a. Security provider defined globally.



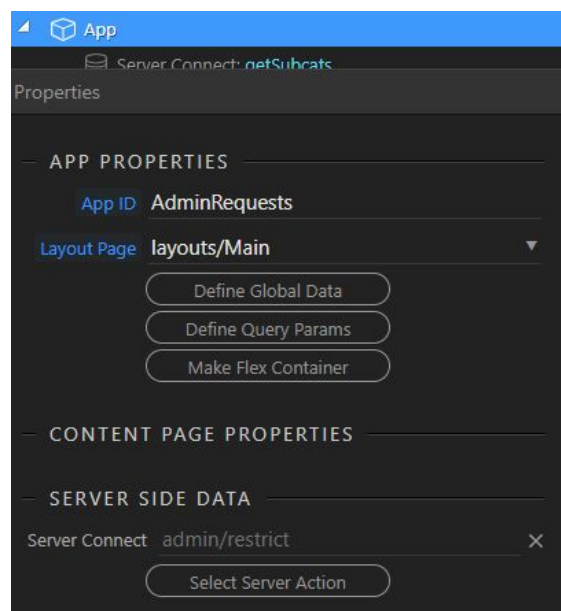
Picture 9b. User permissions established in the security provider.

2. After global definition of admin permission, I created a separate API for restrict action (Picture 9c).



Picture 9c. "Security restrict" API action. Admin users can access admin link ("/").

3. The last step to restrict the page is to establish Server connect on the frontend (Picture 9d). After that the admin page can be accessed by admins only.

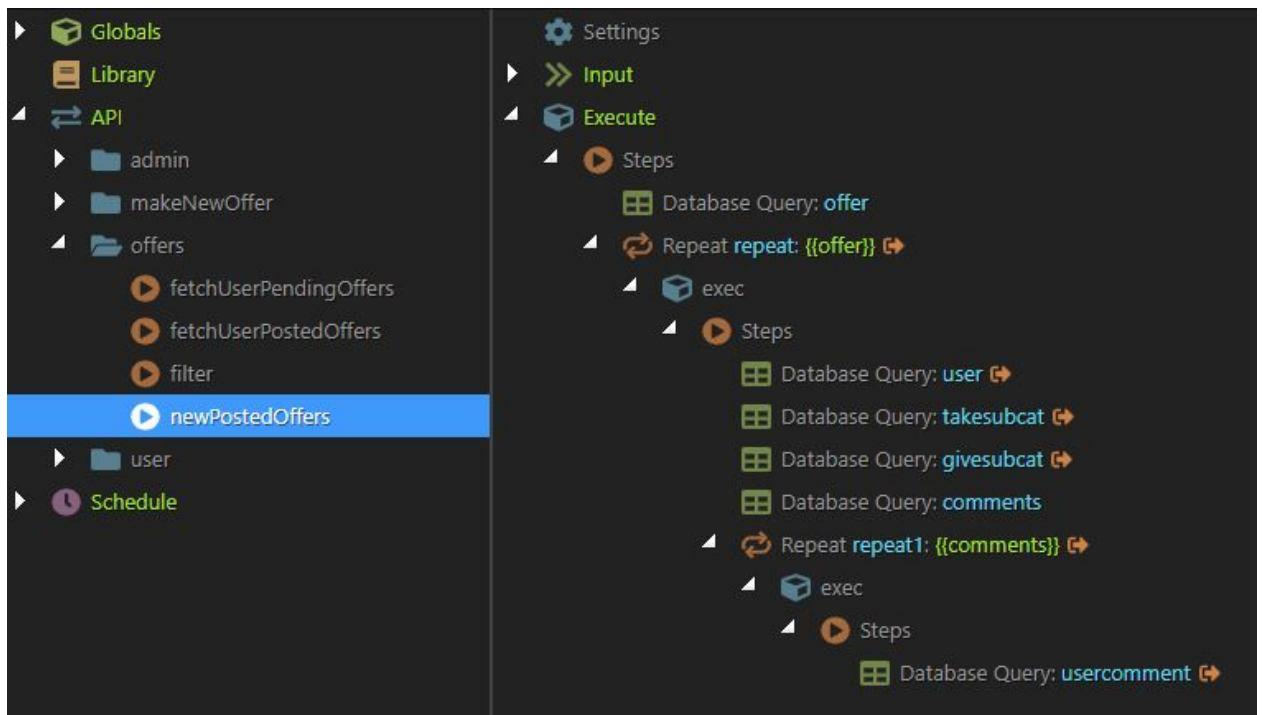


Picture 9d. Server connect action for App of AdminRequests page.

Fetch data

In my project fetching of offers is done more or less in the same way on different pages. This section explains the implementation of fetching new posted offers from all users and displaying them on the Home page.

1. Establish API action with queries. Please refer to Picture 10a.



Picture 10a. Example: newPostedOffers.

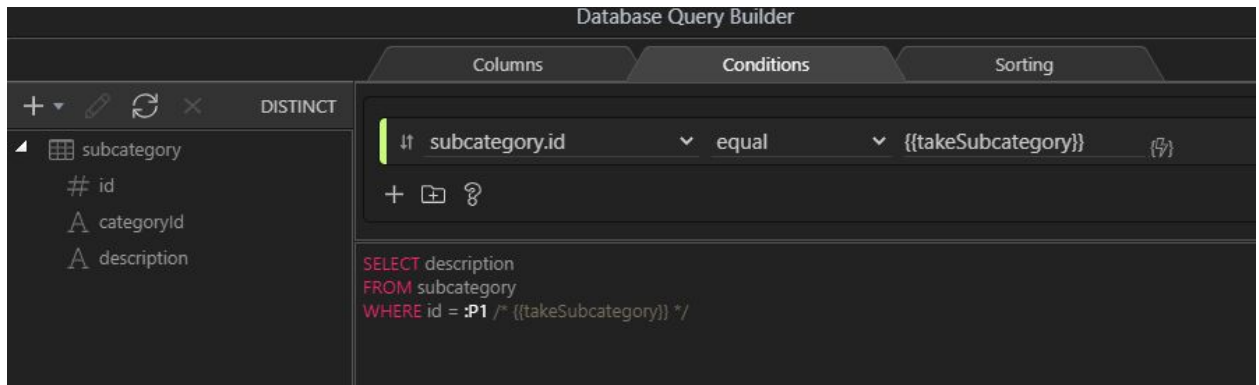
The idea is that for every offer there exists the creator of the offer (user) and comments. For every comment there exists the creator of the comment (user) and the comment itself. That is where the Wappler's Repeat actions come handy.

First, I establish the main database query "offer", that returns all posted offers from all existing users. I use "offer" as an expression for the first Repeat action "repeat", that iterates through the offers and outputs necessary data. For my offer card (Picture 4b), I need to output the following information: name, surname, username and avatar link of the creator (user), "take/give" subcategories and comments.

2. The Offer table in the database stores references to user and subcategories tables. So, in order to get the subcategory description, I need to make sure that "take/give"

IDs of the subcategories match the “take/give” subcategories’ references in the Offer table. This can be done via the Conditions section of query builder. Please refer to Picture 10b.

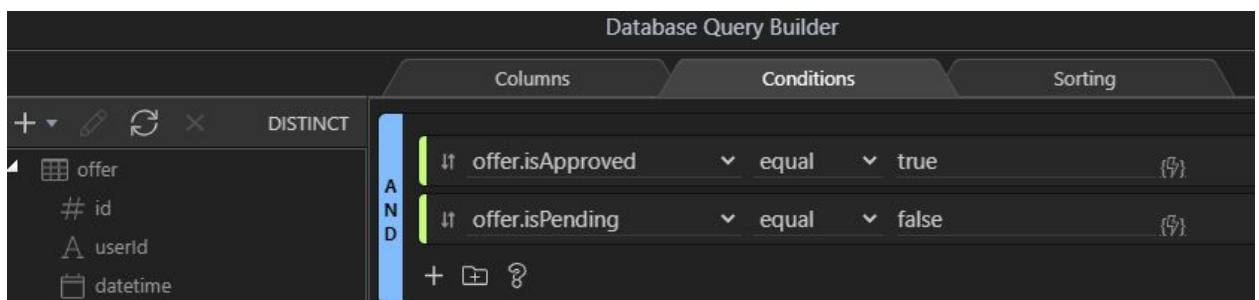
I establish conditions for “user”, “comments” and “givesubcat” the same way.



Picture 10b. “Takesubcat” query.

Same idea goes for comments: “comments” query is used as an expression in Repeat action (“repeat1”). Repeat iterates through every comment and outputs username and avatar of the user (“usercomment”).

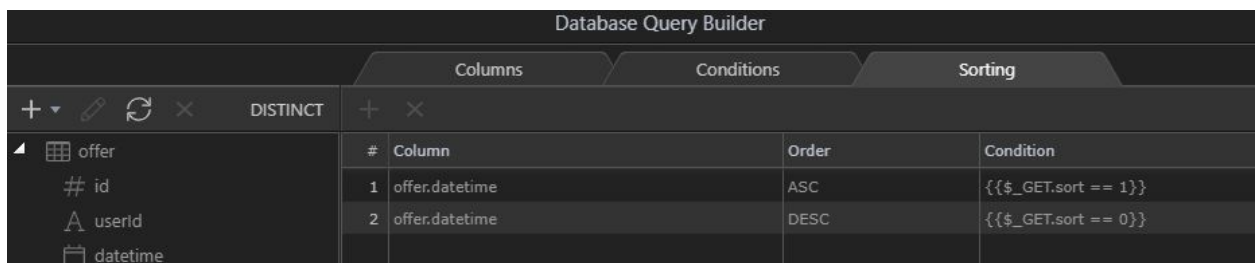
3. So, after setting up the template for fetching, I need to make sure that only the offers that have been posted by users are displayed. According to Figure 3, the following conditions need to hold: offer should be approved and offer should not be pending. The conditions are added to the “Conditions” section in the main query “offer”. Please refer to Picture 10c.



Picture 10c. Conditions for the “offer” query.

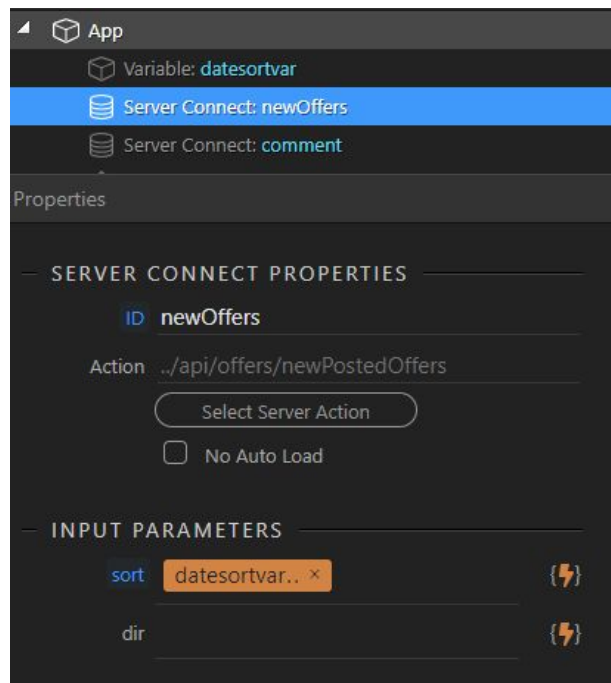
Another feature of Home page is sorting the offers based on the date they were posted. I wanted to achieve that on every odd click of the button the sorting is ascending and on every even click the order is descending. This can be implemented on the backend with the “Sorting” section of the query builder.

4. The “sort” variable is defined as GET input of “offer” query, and when the variable equals 0, the order is descending, if sort is 1, then ascending. Please refer to Picture 10d.



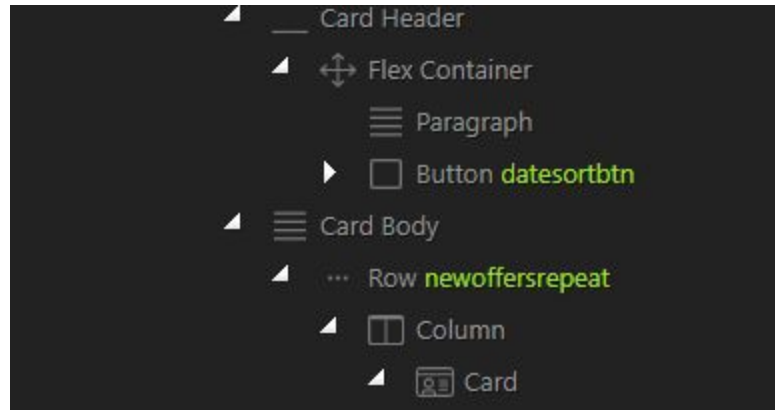
Picture 10d. “Sorting” section of query builder.

5. Now the API action is complete on the backend and next step is to connect it to the frontend. For that the Server Connect action is added on the Home page. For sorting the “datesortvar” variable is initialised as 0. Please refer to Picture 10e.



Picture 10e. Server Connect that uses “datesortvar” for sorting.

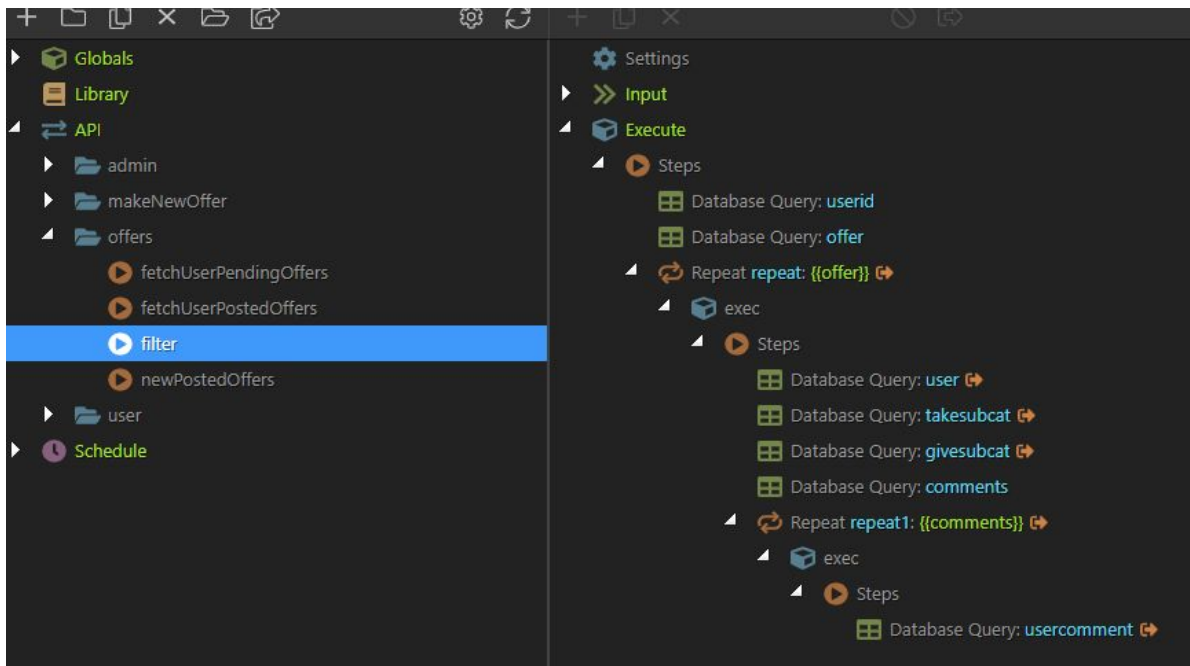
-
6. The button in the upper right corner of the card (Picture 4a), “datesortbtn”, changes “datesortvar” from 0 to 1. This makes sorting of Home page dynamic. On the frontend I make use of the Repeat region. The inner card of the Home page has one row with repeating columns. The row’s repeat properties are connected to the Server Connect repeat action, defined in step 5. Please refer to Picture 10d.



Picture 10c. Repeat properties for a card element to fetch user's pending offers.

Filtering menu

The filtering menu is quite comprehensive and flexible. The idea is that it allows one to search offers no matter how many parameters are entered. It is necessary to make sure that the search result' offers have already been posted. The steps of APIs are more or less the same as for fetching the data (Picture 11a).



Picture 11a. Filter API.

The challenge of the filtering menu is to define correct conditions of the query. Conditions should make sure of the following:

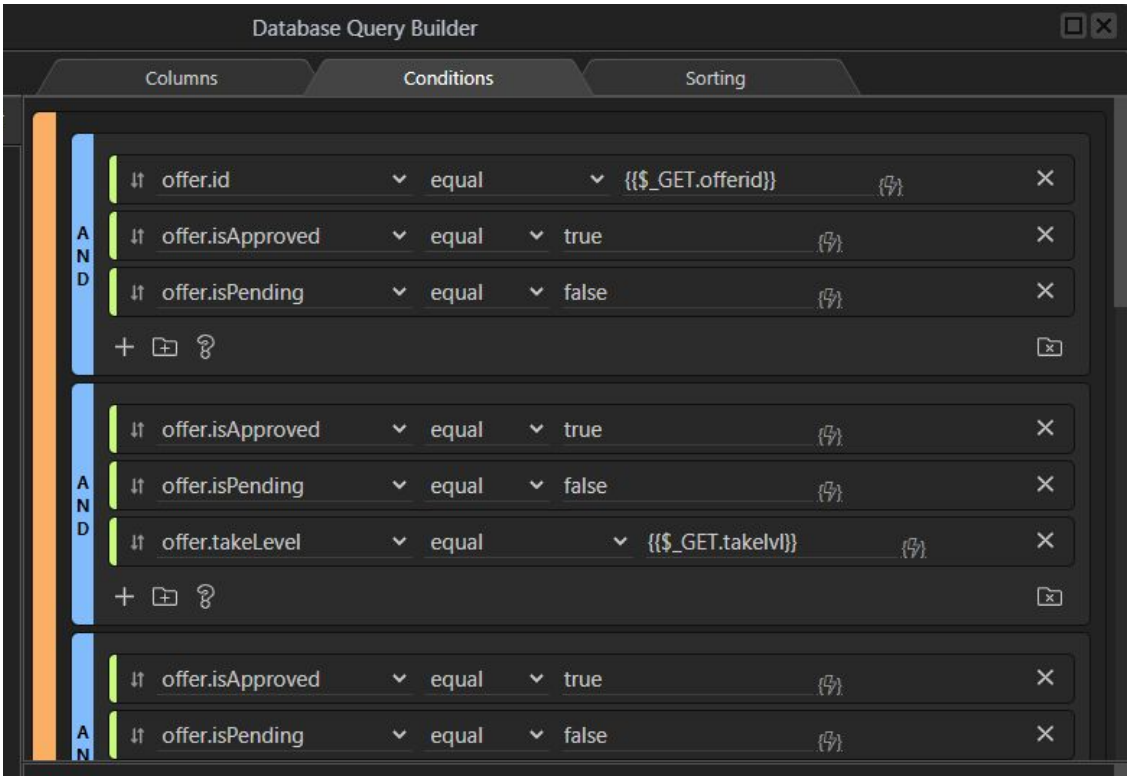
- Users can search with one or more parameters. For example, search should still yield the results (if exist), when the user searches only based on the “give” category.
- All result offers have already been posted.

For that, I added necessary filtering parameters as GET inputs (Picture 11b).



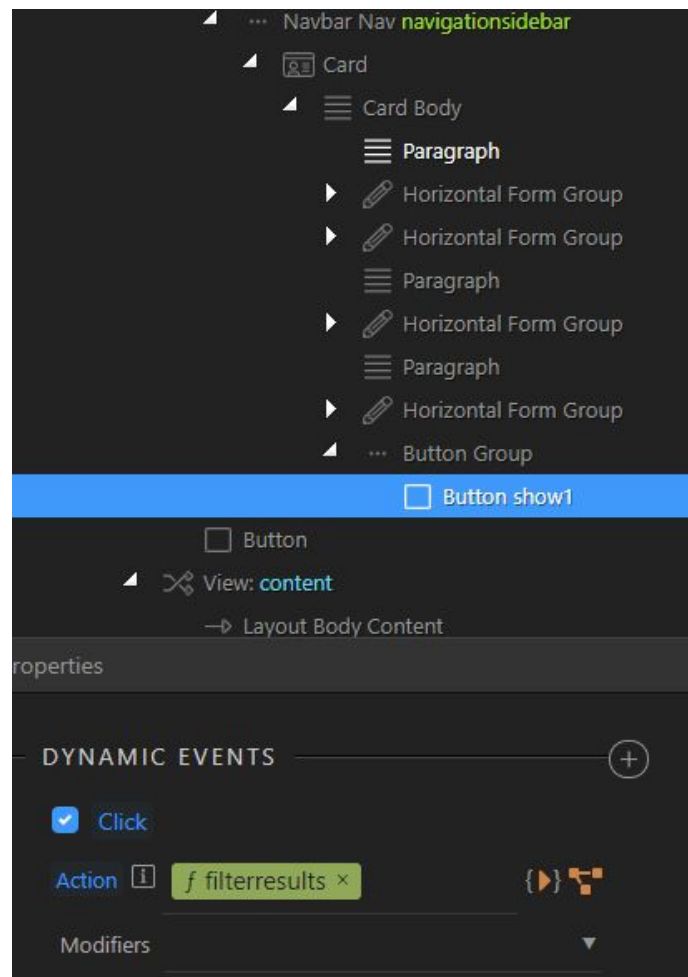
Picture 11b. GET input parameters for filter API.

The next step is to modify conditions of the “offer” query. The idea is that each filter is a separate group of conditions together with offer being approved == true and offer being pending == false. All filter groups are connected via OR gate to make sure that search can be done with one or more parameters. Please refer to Picture 11c.



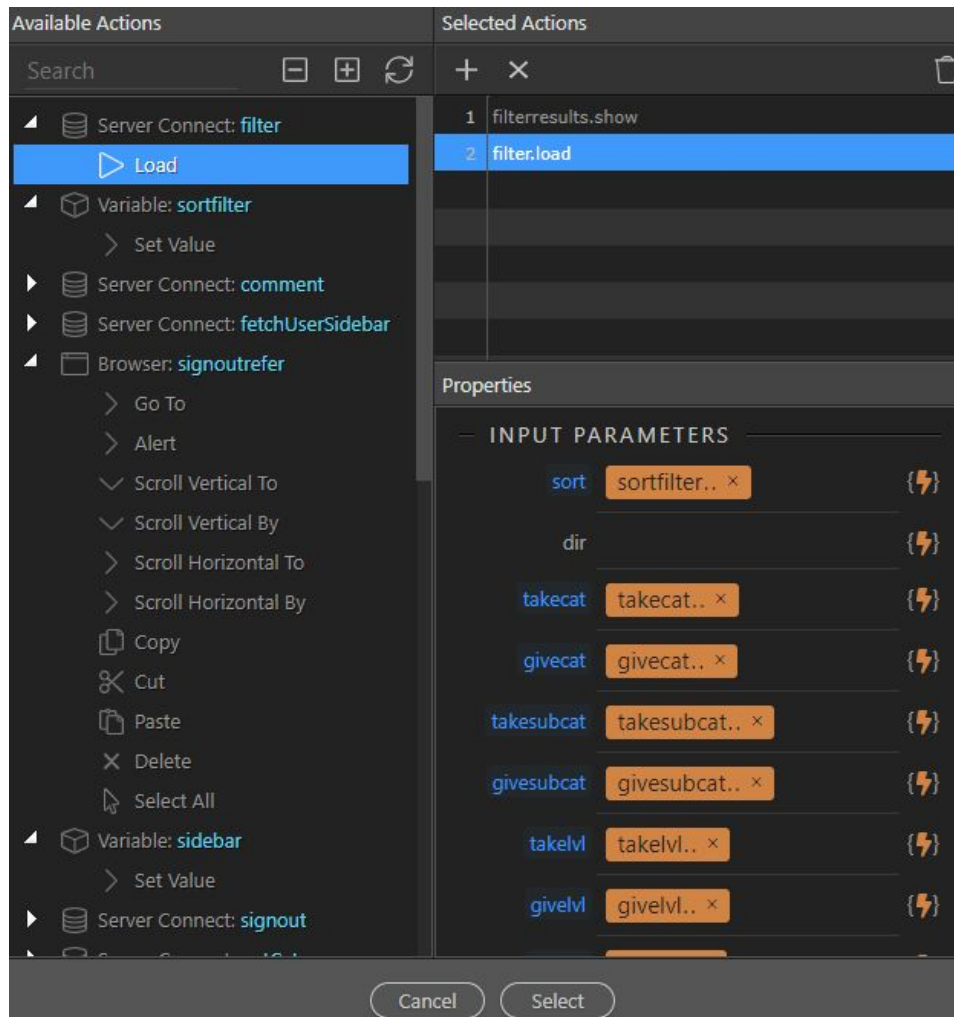
Picture 11c. “Conditions” section of the “offer” query.

Next step is to connect the backend to the frontend. Please see Picture 11d for the structure of the search menu. Here, four horizontal form groups and “show 1” button are defined. The button initiates the Server connect action of filter API.



Picture 11d. Search menu structure.

Inside of the button click action, I first show the modal and then load the Server Connect with needed inputs (GET inputs defined in API), referring to the input fields in horizontal form groups (Picture 11e).



Picture 11e. Show button click action.

The filtering menu is ready to use.

Future development of Skill

At the initial stage of Skill development, I was thinking about what features would be nice to have, or what can be improved and how the concept may grow.

From Technical point of view, Skill may improve the following:

- Search on offers based on user location
- Different GUI modes (Dark mode view, Light mode view, etc.)
- Chat feature. This will enable users to tutor each other right in the web application itself.
- Rating system. Users will rate offers, which will affect the rating of the creators of the rated offers.
- Respond system. For now, users can only comment on offers, however, it would boost user-friendliness, if users could respond to offers to show their interest, and then the creator of the offer would receive notification about the responses.
- Report/block/delete users. As the number of users grows, Skill needs to ensure a safe environment. For that users could report and block other users if they consider the behaviour inappropriate, and if the number of reports is high, the admin may block access of the person or even delete the account.

From Business point of view:

- Bonus programs/titles. For example the user with most responses is “Top taker” and the user with the most responded offer is “Top giver”. This will increase the competitive atmosphere of the application.
- Business account. Companies may register on Skill and represent “I give money - I take skills” principle.
- Self-employed personal account. Self-employed people may present themselves based on the “I give skills - I take money” principle.

As can be seen, Skill is quite a flexible application with a lot of different ways to represent itself both from technical and business aspects.