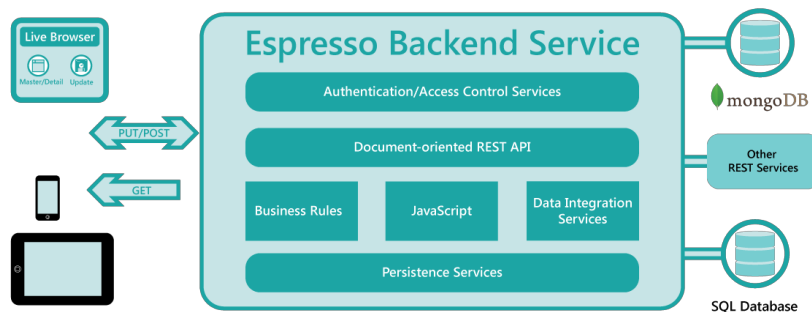


Using Espresso Logic with Appery.io Mobile Platform

Introduction

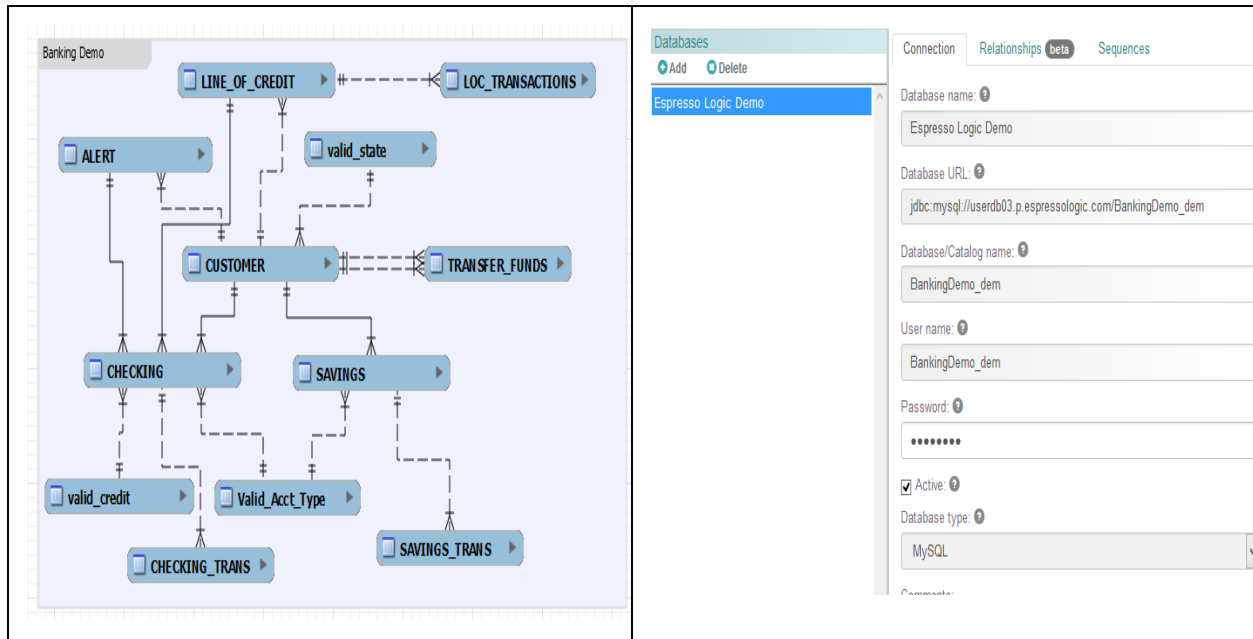
The requirements for delivery of enterprise mobile applications include; a fast and easy-to-use front-end development tool, a secure RESTful integration server, and connectivity to one or more SQL and NoSQL backend data stores. These endpoints should produce document oriented API that closely match application needs - regardless of where the data is physically stored. We also want to put our business logic on the Integration server (not in a mobile client or in multiple back-end systems).

The Espresso Logic integration service platform offers mobile application developers a cloud or on premise suite of tools to build document oriented REST endpoints that connect to multiple data sources. These REST endpoints can come from existing SQL Databases, MongoDB, or other backend systems or REST services. This article will discuss how Espresso Logic created a mobile banking application that allows customer to review personal information, view accounts and transaction details, and transfer funds between accounts. We selected Appery.io as our front-end mobile development tool. The web based UI and drag-n-drop approach to wiring REST requests and responses to fields made this an excellent choice for our customer.

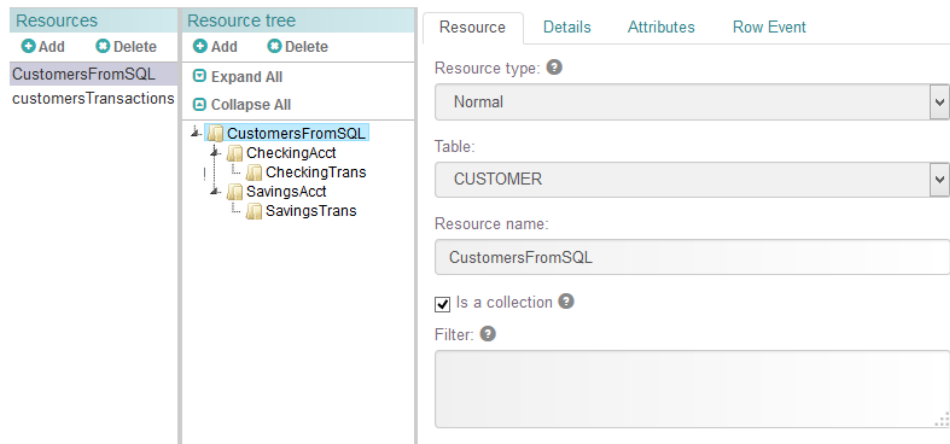


Creating an Espresso Logic Account

[Espresso Logic](#) provides a developer account with a RESTful server and a database in the cloud or you can request a VMWare appliance for local development behind your firewall. Once your account is setup and configured, you can either connect to your own database or use the sample database that is pre-installed with your account. In this sample, we created a data model for MySQL and deployed this to the cloud. In the Espresso Logic Design Studio - we connect our database using the dialog shown below.

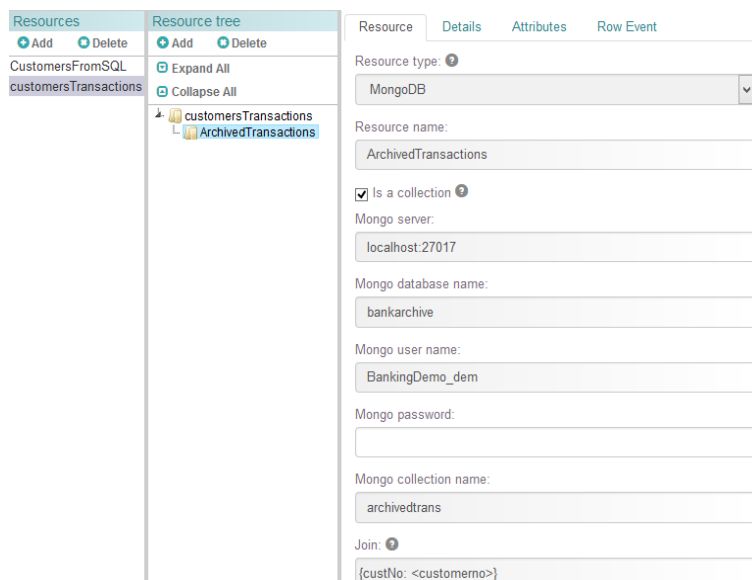


When we connect to the MySQL database. Espresso Logic will build an instant REST API for each of the database tables.



Here we see the Resource Editor in the Espresso Logic Design Studio (a web based tool). Since we are also going to want to show a document style details, we will need to create several new 'resources' to combine SQL to SQL data and SQL to

MongoDB data. In the example above - we see a nested document of all of our customers account and current transaction details which will be used in our front-end.



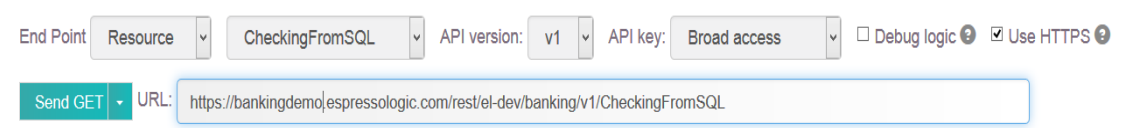
We first define the root resource which is from the Customer table (customerTransactions) from our SQL database. We then add a sub-resource from MongoDB which 'joins' the customer number from our SQL parent to our archived transactions. (For details on the MongoDB integration – see the [documentation](#) page).

One important note – by default Espresso will only return a fixed number

of rows in both the SQL Parent (20 by default) and a fixed number of rows in the MongoDB children. This reduces the risk of flooding the client with too much data. In addition, the REST API will return links for the next batch which can be used directly to retrieve the next set of details.

Setup an Appery.io Account and new project

Once you setup a new appery.io account and create a project you will create several REST service endpoints. Each service is used and mapped individually, that is, you will need one for a GET (read), POST (insert), and PUT (update), as well as DELETE. Espresso Logic has a built-in REST Lab to test your endpoints – so after you have created your project and connected to your database and MongoDB servers you can use the REST Lab to see and test your results. In this example, we are looking at our document 'CheckingFromSQL' as an example. (This returns the JSON in the response window - not shown).



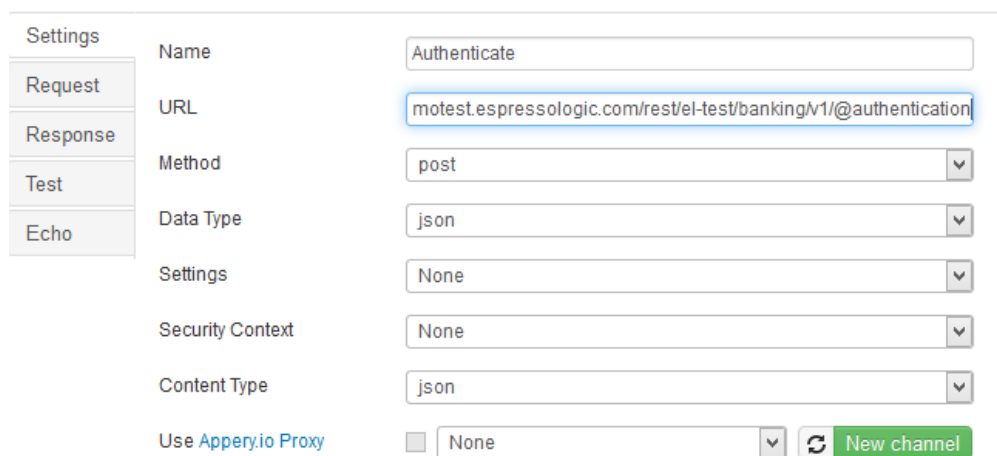
End Point: Resource | CheckingFromSQL | API version: v1 | API key: Broad access | ☐ Debug logic | ☒ Use HTTPS

Send GET | URL: https://bankingdemo.espressologic.com/rest/el-dev/banking/v1/CheckingFromSQL

Figure 1 – Espresso Logic REST Lab

Create new Authentication REST Services

Once we have our REST API – we create a new 'Service' in appery.io project. This special REST Service can be tested using the default API Key created as part of the sample project. (See the Quick Ref in your Espresso Logic for details). The @authentication is a special REST call used to pass the username and password to [your authentication provider](#) and return the API Key that will be used for all future REST calls in the session. Make sure to use **post** for @authentication.



Settings | Request | Response | Test | Echo

Name: Authenticate

URL: motest.espressologic.com/rest/el-test/banking/v1/@authentication

Method: post

Data Type: json

Settings: None

Security Context: None

Content Type: json

Use Appery.io Proxy: ☐ None | New channel

Figure 2 – Create the @authentication service REST Endpoint

We then enter the Request and Response field names shown below.

Name	Default value	Value
username	Demo	
password	Password1	

Figure 3 – Define a Request username and password field names

Name	Value
apikey	
expiration	
email	

Figure 4 – Define the authentication response fields

After the Authentication REST service is added to your logon screen and the request fields are mapped to the input fields and the response value will be mapped and stored in a 'local storage' variable called **apikey** and **errorMessage** to be used on the subsequent REST calls.

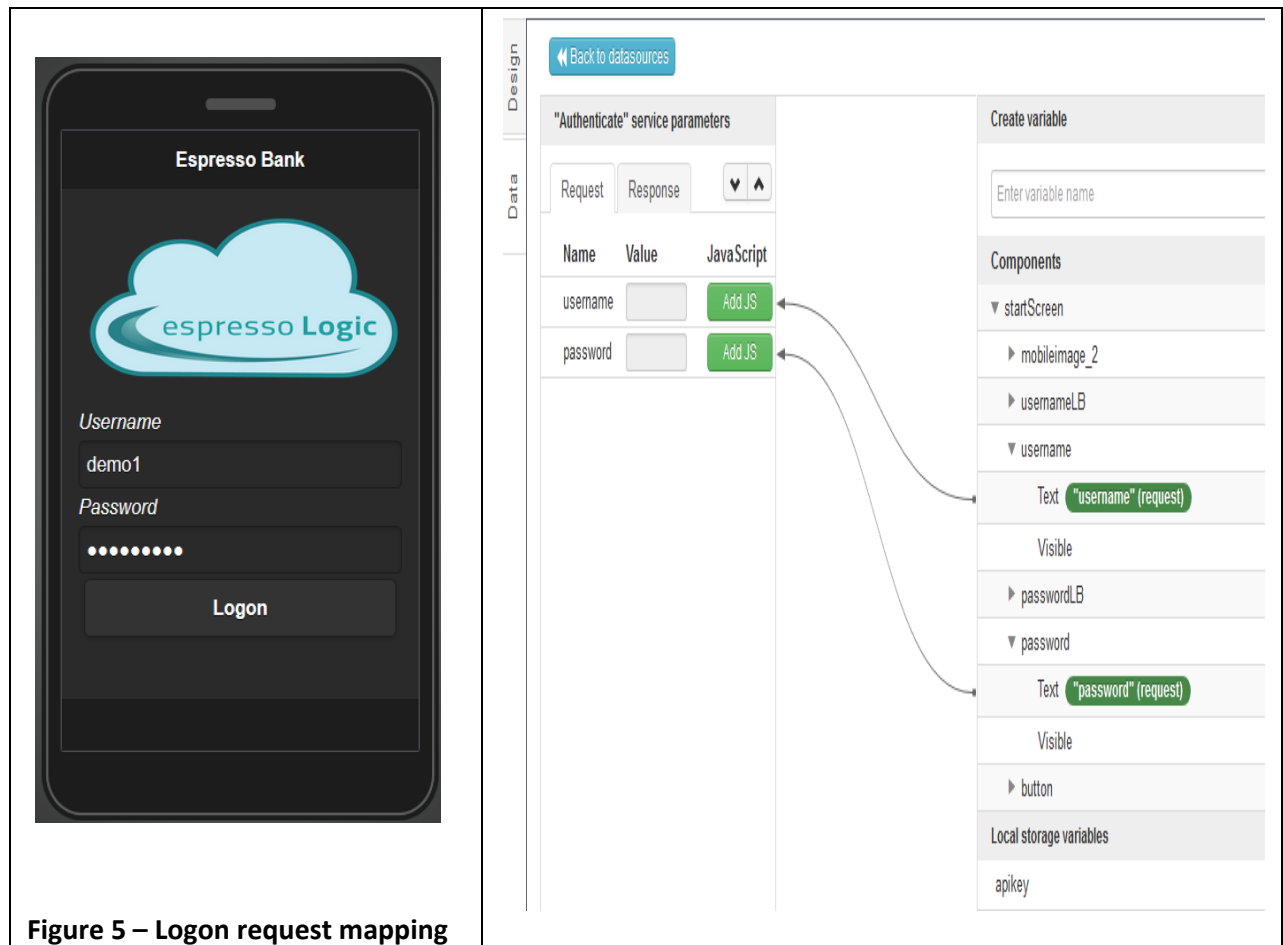


Figure 5 – Logon request mapping

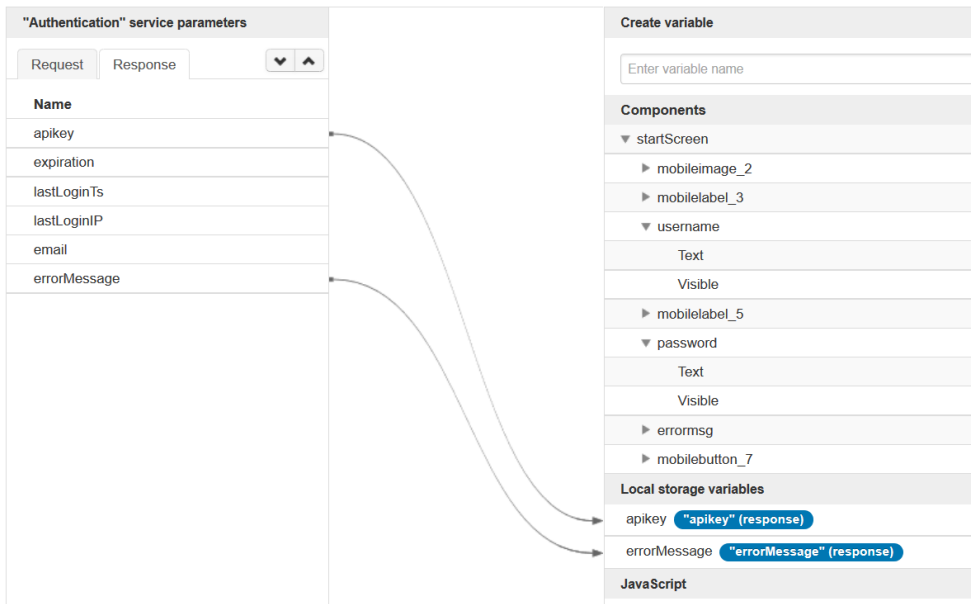


Figure 5- Mapping the authentication response

Events

The next step is to add events to the logon button to invoke the service ‘Authentication’ . If the service is successful we navigate to our next page otherwise we will receive an error that we can display on the logon page.

LogonButton	click	invoke service
Authentication	success	Navigate to CustomerDetails
Authentication	error	Map errorMessage to text field

Figure 6- Events for logon screen

Create a customer detail screen

The next set of screens simply repeat the process of creating and mapping REST GET service calls and mapping fields on the input text fields. In this example we mapped the response array (\$[]) for each field to the corresponding input fields of the CustomerRESTService.

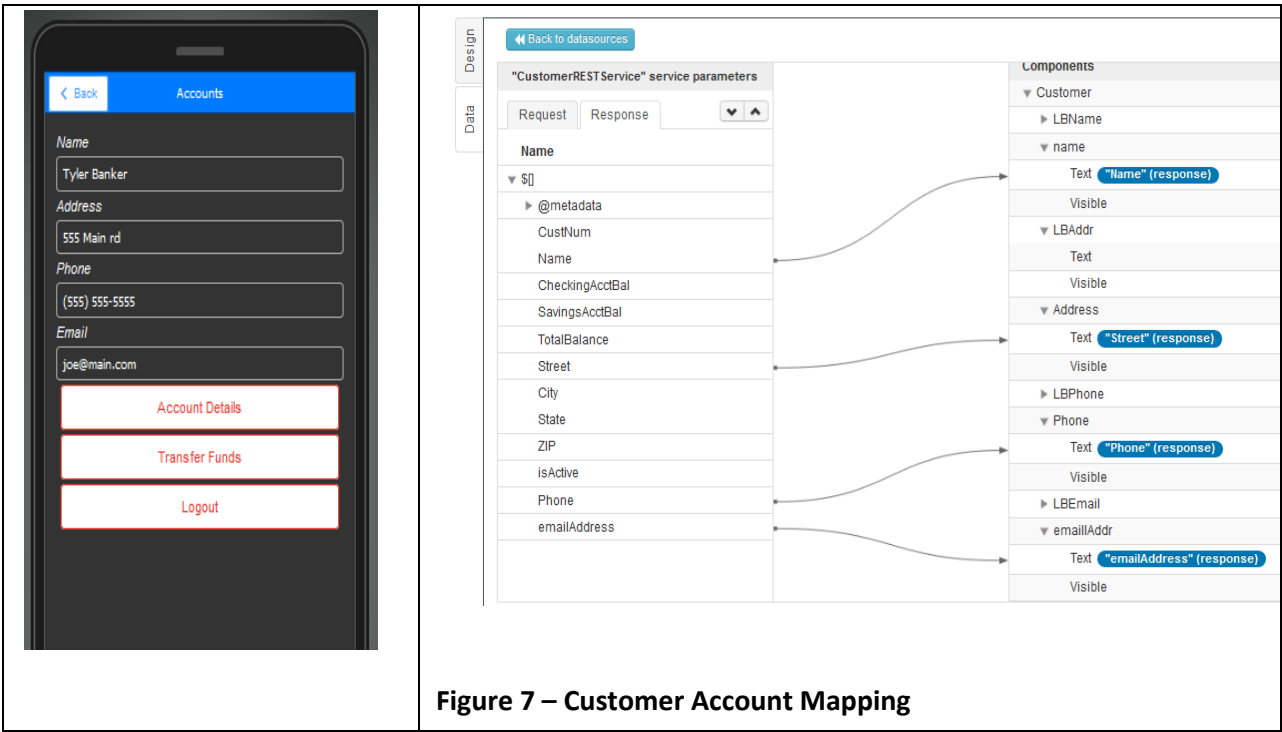


Figure 7 – Customer Account Mapping

A special Authentication apikey is used for testing GET response values - API key: ‘demo_full:1” this is a convenience so you can do a GET from a web browser. We can use this to test our REST calls while we develop our application and security model. This can be added to each of the REST Request fields. This will be replaced with the apikey in the header request for our production application.

Name	Default value
auth	demo_full:1

Transfer Funds

The final step is to create a POST REST call to our transfer funds API and map our input fields to the request. Simply add an event 'Submit' to invoke the TransferRestService.

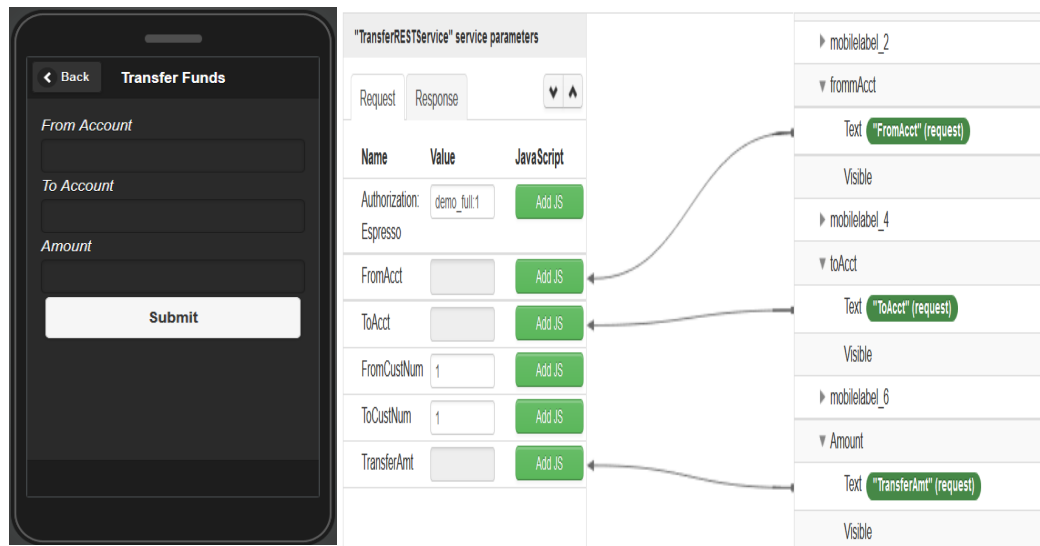


Figure 8 – Transfer Funds Mapping the Request

The appy.io [documentation](#) explains in more detail how to build REST services and deal with authentication.

Business Logic

Behind the scene is a very powerful business logic engine. Think of this as an 'observer' pattern that monitors the state change of data changes on your SQL and NoSQL data. Any POSTS, PUTS, and DELETES can 'trigger' a business rule event and start a chain of actions (sums, counts, calculations, formula, events, and constraint validations to name a few). In our banking model we have rules to calculate the sum of deposits and withdrawals, rules to prevent overdrafts (or trigger line of credit and bank charges), and JavaScript rules to transfer funds between checking and savings accounts. For a quick rules tour – see the Espresso Logic [documentation](#) page.

Security

Espresso Logic also provides advanced authentication to a wide range of corporate security including (Windows AD, LDAP, Stormpath, etc.). The fine-grain role-based access control will determine which role can access the resource (down to the row and column level). Finally, business logic services which can be created to enforce validations, calculations and lifecycle event services. This security extends to new our document oriented resources as well.

Summary

The appery.io interface is a web based drag-and-drop with many slick features to connect and build a mobile application connected to a REST integration server like Espresso Logic. The learning curve is fairly low for simple applications and they 'generate' code behind the scene that is completely accessible. The **'test'** emulator runs in the browser and the QCR tag will let you download and run the application immediately on your mobile device. Using Espresso Logic as a RESful integration server with Appery.io is the fastest way to build and deliver enterprise results. Espresso Logic and Appery.io is a great combination for a successful and agile delivery of any mobile application.

These sample applications for both apperyt.io and Espresso Logic can be found on [GitHub](#).