Atlas Forms Manual (Draft WiP)

Authors: Luce Carter, John Page
Date: 25 October 2022

Introduction

Atlas Forms (AF) is an Open Source project licensed under the Apache 2 license and available at https://github.com/mongodb-developer/AtlasForms. It is not officially supported by MongoDB but is intended to be a starting point for your own application development. The best place to get assistance is at the MongoDB developer forums (https://www.mongodb.com/community/forums/) .

Atlas forms provides a form-based, non-technical end user interface to allow users to Create, Find, Update,and Link Records. It looks similar to a Fourth Generation Language (4GL) in that it abstracts away all database development initially and provides a zero code solution to building a GUI application to interact with data held in MongoDB Atlas.

It differs from a legacy 4GL in having no proprietary coding language - it is instead a skeleton application in modern Javascript built with serverless functions running on MongoDB's Atlas Developer Data Platform. It is intended to be modified to fit requirements and has been designed to be modular and easily modified. It can be run on the Free forever Tier for Atlas, subject to limitations noted in the installation section.

Installation

See: **http://mdb.link/AtlasForms**

Features and usage guide

User Guide

Registering as a User

If this is your first time using Atlas Forms then you will need to register your email as a user. At the Login screen click 'New User' and enter your email address (It must be a valid email address) and the password you wish to use to log in.

You will receive an email asking you to verify this is your correct email address, once you click the link in the email your account will be created and you will be able to log in with the password you just chose. If at any point you forget your password you can click 'Forgot password' on the login screen to reset it.

After you log in you may or may not have permissions to do anything useful in the system, this will be determined by your administrator - you may need to ask them to grant you appropriate permissions or they may have configured the system to give you appropriate permissions automatically. If you are the person installing the system then the first user will have superuser permission.

The Form Screen

Most user functionality in Atlas Forms is accessed through a single screen. This screen is divided into three parts vertically.

At the top of the window is the Menu Bar, this is where you select the form type you want to work with, enter full text searches, and click to perform actions like **Search** or **Create** data. The **Logout** button is also located here at the top right.

Underneath the menu bar is the Results View. This shows you a summary of all records matching your latest search, you can scroll this list. You can also sort by any column by clicking on the column header and you can resize the columns by dragging the bottom right corner of the column name box indicated by three small diagonal lines.

The lower part of the window is the Form View where you can enter and view data, this will attempt to optimally fill the available space as you resize the window. You can drag the bottom right corner of any text box to make it larger to allow you to view more data without needing to scroll.

Selecting a Data Source

Choose which form you want to work with from the drop down list at the left side of the Menu Bar. This will show all data sources your administrator has made available to you.

Querying By Form

You can use the form to search for data by entering data into boxes that define what you want to find then clicking **Search.** You can enter values in multiple boxes and all these must exist in a record for that record to be found and shown to you. You can also enter greater-than (>) and

less-than (<) symbols in boxes, for example entering >2 in a box will find all records where the value in that field is a number larger than 2.

Text Search

If it has been enabled by your administrator you can type any word or multiple words into the Search Box on the Menu Bar and click search, this will match any record where one or more of those words, or similar words exist in the documents. The exact definition of what records match a given query is controlled by your administrator.

Filtering Results

When you click Search, whether using the Search Box or entries in the form you will be shown a summary list of results, if you wish to refine this list then, *before clicking any result,* you can add additional criteria to the form or text box and click the Search Button again to narrow down the results. Once you view a result you will not be able to modify your search until you click Clear and re-enter the search criteria.

Field types

Some fields on the form may have drop down lists offering a range of appropriate values, you should use these rather than typing an answer where offered.

Some fields will only accept numbers.

Dates are shown as date choosers, of you do not know the time you should leave it as midnight

Multi value fields

Some fields allow you to specify multiple values both when saving data and when searching. You can see these by the add (+) and remove (X) icons by the value, use these to create the required list of values.

Clearing a form.

To clear the form, and enable a new search or create press the **Clear** button.

Creating a New Record

To create a new record simply fill out the form and click **Create** , if there is an error, for example you fail to add a field your administrator requires then you will be shown an error but you can edit and resubmit.

Editing a Record
If you are viewing a record on a form and wish to make a change to it, you can click the **Edit** button. As long as no-one else is currently editing it then the fields will become editable. When you have completed your changes click Save changes.

Administration User Guide

Atlas Forms divides users into three categories: users, superusers and developers. A superuser explicitly has all permissions where a user must be granted permissions explicitly. The first user created is by default a superuser.

A developer refers to someone with access to the Atlas application services backend and can redefine the system behavior.

Any user can be granted permission to manage Users and permissions, Available Form types and the contents of the drop down picklists.

All administration tasks are performed using the same user interface with special forms for Users, Forms and Picklists available in the Form type chooser.

Managing Users

When a user first signs up they will only have the permissions defined as a default for them by the Developer. By default this gives no permissions to anyone beyond the initial user. Therefore users must be explicitly granted access to data sources.

Authorizations

Authorizations are specific permissions allowing a user to perform action. The default configuration supports the idea of read, create and edit as separate permissions for each form type with read required for create and edit.

You can grant authorization to a user by selecting the form type AF_Users and then adding the required authorization to the permissions list. These can be a comma separated list or individual permissions.

The permissions you can grant are:

Permissission	Meaning
READ	Required to see the existence of a document type / form
CREATE	Required to make a new record.
EDIT	Required to Edit a record
READDOC	Required to Query and view records,. Called Once for every record in a result set

These can also be granted to the AF_Users, AF_Picklists and AF_Doctypes collection if you wish to give a user limited access to these but not grant them superuser.

Additional rules and permissions can be defined by a Developer, for example limiting which fields can be seen or modified by a given user or group of users.

Managing Doctypes

The list of available forms, what collections they point to and how they are displayed are managed using the AF_Doctypes form. By adding a new record to this collection you can allow users access to a new set of data. The Doctype record does not define security, access must be explicitly granted. Either in a user's record or programmatically by a developer.

When you first and a document type you do not need to complete the schema box, the list of fields, their data types and typical sizes will be auto discovered the first time someone logs in with access to that datatype.

Once the schema has been discovered you can edit it to change the ordering of fields or data type information. The schema is used to ensure correct data typing and to define what the form looks like.

The schema takes the form of a JSON object with the same set of fields/keys and shape as the data. The values contain a string representing the data type, for string fields this can also be "string:Size" where size is a maximum typical size in characters. This is used to determine the width of the box on the form

Atlas Forms only renders a maximum of 2 levels arrays in objects or objects in arrays beyond this is hard to present on a form and the rendering code is not designed to handle recursion. you should limit the schema to ensure it does not try to render deeper than this.

Importing Data

When in the AF_doctypes form you can click the import button to move to the import module.

The import module is intended for learning and testing rather than as a production data import mechanism, for production imports please look at other MongoDB tools.

Import allows you to point at a JSON source on the internet, a file or public API, and import it to Atlas to create a new data source and associated form.

Managing Picklists

Drop down lists of value choices for fields are known as Picklists and are managed via the AF_picklists data type. Each record in this Collection specifies a database, collection and field as well as a list of acceptable values. If a Picklists record exists then it will be shown for that field

Generating Picklists

A developer can auto generate a Picklists based on a sample of the data in the database, there is no import feature at a User or Superuser level.

Developer Guide

Introduction

Atlas Forms has been designed to work with little to no additional development, but to be easily understood, adapted and modified to become a more custom application for your organization. Car has been taken to comment any code that may be unclear and you can always reach out for assistance on the MongoDB Developer Forums

It has a plug-in architecture for back end authorization and verification which means you can enforce business logic and rules by writing your own functions without needing to change any existing code. How to do this is explained later.

Architecture Overview

Front End Rendering

The front end rendering code is written using Vue.js and Bootstrap delivered via a CDN.

Vue is a relatively un-opinionated frontend framework and Atlas Forms deliberately uses only a simple subset of Vue functionality to make it easy to understand. If you are not familiar with Vue.js - Vue was chosen over Angular and React as the simplest with a frontend-only, uncompiled, real-time rendering model, therefore the easiest to modify should you wish to do so.

Vue is used from a Content delivery network (CDN) and there is no compilation / transpilation phase required, changes you make to front end code are immediately displayed in the output. Each page of the application (Main GUI, Import, Login, Signup, Password change) is a separate html file and supporting js files in its own directory that renders that page.

Using Vue you create a VueApp in Javascript and define functions (in JS) and data members for that - the web page can then use these values and functions when rendering, to display values, to iterate over values and to call functions for formatting and conditional display logic. The Form page is documented in more detail later.

Bootstrap was chosen as a standard CSS framework, again for simplicity - additional CSS is added to assist with some rendering.

Front End Communications

The browser frontend communicates with the Atlas Developer data platform using the Realm-web SDK, this is the simplest and most secure way to expose server side Atlas functions. The front end makes no explicit HTTP calls and no callable REST services are exposed. The Realm SDK is available for browsers and many mobile platforms. You can choose to expose functionality as authenticated REST if you need to for another service.

Atlas Forms uses only functional API's , not data APIs as a security best practice this is described below .

Back end serverless functions

All functionality is defined by a small number of client callable business level functions which perform a specific application action, for example getting the list of document types a user can see. This contrasts with opening up data access APIs directly to the applications, database level APIs which allow the client broad access to choose the data it wants can be a security issue.

That sort of Data API should never be provided to an insecure (end user) environment as this ability to access a set of data not strictly controlled by the server is often a cause of data breaches.

In addition an API should where possible where isolated from the data model. So you get the set of data the frontend needs without needing to understand how that is stored on the server. This allows you to evolve your data model.

Finally, as you will see if you read the supplied serverless functions, some operations which seem simple from a client perspective may have additional technical, security and optimization steps that should not be required to be implemented by all frontends.

To this end Atlas Forms defines a number of serverless functions encapsulating high level behavior like getSchema or queryDoctype or lockForEdit which are a defined specific purpose API, these are then called by the client using the Realm SDK.

Authentication

Atlas Authentication supports a number of different ways for end users to authenticate themselves, from anonymous but distinct authentication based on the web browser and cookies through email/password to social media (Facebook/Apple/Google) authentication as well as JWT Tokens and federated authentication via a custom function.

Atas forms has been written to use email/password authentication as that is simple, and generic however it would not be difficult for a developer to plug in a different authentication mechanism as required.

Authorization

A key feature for any end-user system is authorization, determining what actions a user may take. Atlas Forms uses a simple system of positive authorization for actions. Each user must be explicitly granted rights to read, create and edit on a given data source. In addition to this every action performed by a user will be passed to a custom function if that function has been defined it can allow or deny the action, choose what message to present to the user and also modify the action as required.

For example when creating a record this could be used to enforce mandatory fields, explicitly populate fields or warn a user about incorrect usage. It can also be used to filter query results either removing entire documents - or removing/redacting specific fields.

Although the Developer Data Platform includes Atlas Authorization to apply rules to the low level database APIs - in our case we want to allow this to be written (and tested) as first class Javascript code and to apply security not to the underlying database interactions but rather to the larger scale business functions we present. More details of how to use this appear in the next section of this guide.

Implementation Details

List of Functions

Name	Visibility	Description
ADMIN_PicklistGenerator	Private	Can be explicitly run by a developer via the CLI/GUI/Admin API to generate a picklist for a given form and field combination.

commitEdit	Public	Called to complete a given edit operation and unlock the document. Can also be used to cancel an edit
createDocument	Public	Called to create a new document
getDocTypeSchemaInfo	Public	Called to retrieve Schema information for a given form
getListOfDoctypes	Public	Called to retrieve a list of Form a user has access to
getPicklists	Public	Used to retrieve all the defined drop-down picklist information for form.
importDocType		
internalCommitEdit	Private	Called by both commitEdit and createDocument to apply requested changes.
lockDocument	Public	Called to express a desire to edit a document, if it succeeds then the document is locked by the calling user.
newAuthorization	Private	Returns an Authorization class, this is called at the start of each public function to get the class that can be asked for security decisions.
newUserSignup	Private/Trigger	Called by a trigger when a new user signs up
queryDocType	Public	Performs either a MongoDB Query or an Atlas Forms Query or Both to return a set of results.
sample_custom_verify	Private	Skeleton Authorization function to use in developing your own
utility_functions	private	Small functions used by other classes such as enforcing data typing or determining BSON type of a JS object.

verify_EDIT___atlasforms_doctypes	private	Custom authorization function for __atlasforms.doctypes, verified that the schema is valid JSON for example.

List of Authorization Hooks

Defining an Authorization / Verification Hook

Authorization hooks are called at multiple places in the code - they are called through the authorization class which is instantiated at the start of every frontend accessible function call. To implement an authorization hook create an Atlas Function with a name of the form verivy_ACTION_DOCTYPE(user, document, changes, ... additional) , this will be passed the parameters for the required operation, the document or the id of the document it applies to it applies to and allows these to be modified. For write operations you will have the _id of the record to be modified and the list of changes, for a read operation you will have the contents of the record.

The list of actions that can be overridden are.

Permissission	Meaning
READ	Required to see the existence of a document type / form
CREATE	Required to make a new record.
EDIT	Required to Edit a record
READDOC	Required to Query and view records,. Called Once for every record in a result set

You can use the Duplicate Function feature in Atlas Functions to copy the example authorization function. An Example is also given below of **verify_READDOC_sample_airbnb_listingsAndReviews**


```
function verify(grant, targetRecord, ...args) {
 grant.granted = true; //Allow everyone to see this document
 grant.message = undefined; //No message shown for READDOC
 targetRecord.price = "" ; //Remove the price value

}

exports = function () {
 //Return the function, don't call it.
 return verify;
};
```

Generating Picklists

Atlas forms can generate a picklist from the list of values in a field by explicitly running the function
ADMIN_PicklistGenerator in the Atlas Functions GUI - modify the function to set the doctype and field you want to generate a list for. This creates a list if there are 255 or fewer distinct values and there are at least 4 times as many documents as the number of unique values.