Integrate the ToDo App with Twilio

The following tutorial builds upon the procedure in Building a Web-based To-Do App and integrates the `ToDo` app with Twilio. With this integration, you can send messages to your Twilio number to add items to the ToDo list.

Prerequisites

For this app, you need the following:

- The `ToDo` app set up according to Building a Web-based To-Do App.
- A Twilio Account. If you do not have a Twilio account, create one before starting this procedure. You must also create a Twilio project for use with this procedure.

Procedure

A. Go to your `ToDo` app in MongoDB Stitch

Estimated Time to Complete: ~2 minutes

Log in to your Atlas account account and click **Stitch Apps** in the left-hand navigation. Click on the name of the MongoDB Stitch application you created for the Building a Web-based To-Do App tutorial.

B. Configure the Stitch MongoDB Service for Twilio Integration

Estimated Time to Complete: ~10 minutes

You must specify a namespace for each collection the Stitch MongoDB service accesses. A namespace is a combination of the database name and the collection name. In Stitch, each namespace contains rules and filters that control access to documents during read and write operations. Stitch creates basic rules and filters for the namespace when you create it. See the rules section of the MongoDB overview for more information.

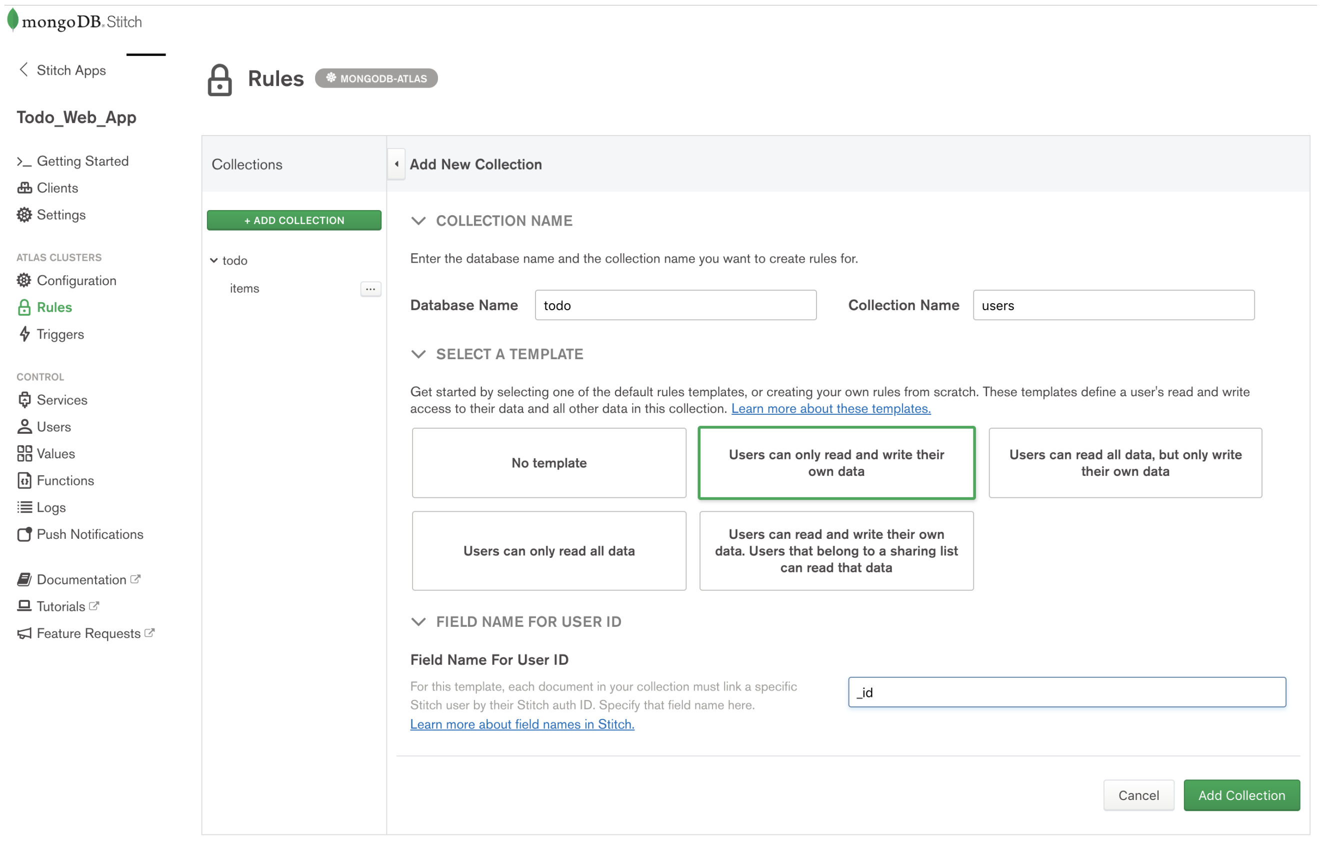
1 Add the `todo.users` collection to MongoDB.

a. Click on `Rules` under **Atlas Clusters** in the left navigation pane.

b. In the **Rules** editor, click **Add Collection** to add a new namespace. Enter the following information:

- For **Database Name**, enter `todo`.
- For **Collection Name**, enter `users`.
- Under **Select a Template**, select the `Users can only read and write their own data` template. The **Field Name for User ID** section displays below the list of templates.
- Enter `_id` for the field name.

Your configuration should look like this:



COLLECTION NAME

Enter the database name and the collection name you want to create rules for.

Database Name: todo
Collection Name: users

SELECT A TEMPLATE

Get started by selecting one of the default rules templates, or creating your own rules from scratch. These templates define a user's read and write access to their data and all other data in this collection. Learn more about these templates.

No template	Users can only read and write their own data	Users can read all data, but only write their own data
Users can only read all data	Users can read and write their own data. Users that belong to a sharing list can read that data	

FIELD NAME FOR USER ID

Field Name For User ID

For this template, each document in your collection must link a specific Stitch user by their Stitch auth ID. Specify that field name here.
Learn more about field names in Stitch.

_id

Cancel | Add Collection

click to enlarge

c. Click the **Add Collection** button at the bottom of the editor.

C. Configure the Stitch Twilio Service

Estimated Time to Complete: ~10 minutes

NOTE:

This section assumes you have a Twilio account *and* that you have created a project to use with this procedure.

1 **Add a Twilio service to your `ToDo` app.**

a. Click **Add service**.
b. Click **Twilio**.
c. For the **Service Name**, enter `tw1`.
d. Enter your Twilio Account **SID** and **Auth Token**. You can find these values in the Twilio dashboard for the Twilio project you are using for the `ToDo` application.
e. Click **Add service**.

2 **Create an incoming webhook to the `tw1` service.**

A MongoDB Stitch **Incoming Webhook** is a callback URL used by third- party service providers to execute a Stitch function. The function creates a document and inserts into the `todo.items` collection.

To create an incoming webhook for the `tw1` service, do the following:

a. Click the **Incoming Webhooks** tab.
b. Click **New Webhook**.
c. Configure following properties for the webhook:

Property	Value
Name	`twilioWebhook`
Respond With Result	Toggle the switch to enable this property.

3 **Configure the function for the `twilioWebhook` webhook.**

Hover over the webhook function and click **Edit**. Modify the function as follows:

1. Copy the following code into the text entry box:

```
exports = function(args) {
  const db = context.services.get("mongodb-atlas").db("todo");
  const users = db.collection("users");
  const items = db.collection("items");

  users.findOne({ phone_number: args.From }).then(user => {
    const todo = { "text": args.Body, "owner_id": user._id };
    return items.insertOne(todo);
  });
};
```

   The function above queries `todo.users` to find the user associated with the originating phone number of the text message. The callback inserts into `todo.items` a document with the user's `owner_id` and the text in the message. `args.From` and `args.Body` correspond to the parameters of the Twilio message.

2. Click **Done**.

4 **Configure the function to send verification codes.**

In the left navigation pane click **Functions** and then **New Function**.

Enter the following properties for the function:

Property	Value
Name	`sendConfirmation`
Private	Leave unselected since this function is called by the client.
Can Evaluate	Leave empty.

In the **Function Editor** set the function code to the following.

1. Copy the following code into the text entry box:

```
exports = function(recipient, code) {
  const twilio = context.services.get(
  const ourNumber = context.values.get("ourNumber");

  twilio.send({
    from: ourNumber,
    to: recipient,
    body: `Your confirmation code is ${code}.`
  });
};
```

The function above sends a text message to `recipient` containing a verification code. It is called from the `todo` web client as follows:

```
let code = generateCode(7);
this.props.stitchClient
  .executeFunction("sendConfirmation", this._number.value, code)
  .then( /* ... */ )
```

2. Click **Done**.

5 **Configure `tw1` service rule.**

Configure up a rule to enable the send action.

a. Click the **Rules** tab.

b. Click **Add Rule**.

c. Set up the following rule which allows the service to send a message to any number.

Actions	When
Enable Send.	`{ }`

d. Click **Save**.

D. Create a Twilio Programmable SMS Service

Estimated Time to Complete: ~5 minutes

1 **Log in to your Twilio account.**

2 **Open up the** Programmable SMS **dashboard.**

From the Twilio dashboard:

a. Click **All Programs And Services** in the left-hand navigation.

b. Click **Programmable SMS**.

3 **Configure** Programmable SMS**.**

a. Click **Messaging Services**.

b. Click **Create new Messaging Service.** Enter a **Friendly Name** to describe the messaging service. For **Use Case**, select **Mixed**.

c. Check the **Process Inbound Messages** checkbox. For **Request URL**, enter the `Incoming Webhook URL` you created when configuring the MongoDB Stitch Twilio service.

4 **Configure** Programmable SMS **Phone Numbers.**

a. Click **Numbers** in the left-hand navigation bar for **Programmable SMS**.

b. Select a Twilio number for the service. If you do not yet have a Twilio number, the **Programmable SMS** dashboard prompts you to **Buy a Number**.

c. Click **Add Selected**.

5 **Configure** Verified Caller IDs **for your Twilio number**

a. In the left-hand navigation, click **All Products and Services**, then click **Phone Numbers**.

b. Click **Verified Caller IDs**.

c. Click the plus-sign and add a phone number that you will use to send messages to the Twilio number.

For more information on the Twilio Programmable SMS service, including features and pricing, visit twilio.com/.

E. Add your Twilio Phone Number as a Stitch value

Estimated Time to Complete: ~2 minutes

Values are named constants that you can access in Stitch rul

do the following:

1 **Click** Values **in the left-hand navigation.**

2 **Enter** ourNumber **for the** New Value Name**.**

3 **Enter your Twilio number as the value.**

Enter the Twilio number you selected while Creating a Twilio Programmable SMS Service as a E.164 formatted string. For example, if your Twilio number is (222) 333-4444, enter `"+12223334444"` (note the included quotation marks).

4 **Click** Save**.**

F. Set Up and Run the ToDo Application

Estimated Time to Complete: ~5 minutes

1 **Run the `ToDo` application.**

If you are not running the `ToDo` application, start. Run the following command from the `todo` directory:

```
npm start
```

2 **Associate a phone number with your login id.**

1. Open a browser to `http://localhost:8001/` to access the ToDo app. Sign in with Google or Facebook credentials, depending on which you have set up in the Building a Web-based To-Do App tutorial.
2. Click `Settings`.
3. In the `number` box, enter a phone number from which you will send texts to your Twilio number.
4. Press the `Return` key.
5. Upon successful messaging, a `verify code` box appears.
6. Enter the verification code sent to the phone number you entered.
7. Press the `Return` key.

Upon verification, messages sent from the phone number to your Twilio number associated with your messaging service will show up as items in your todo list.

3 **Add todo items via Twilio.**

1. Click on the `Lists` link to return to the `ToDo` items list page.
2. Send an SMS to your Twilio phone number.
3. Refresh your browser to see your message.