**Architecture and Updated Project Plan Report**

Ice Cream Tracker

Neapolitan Solutions

Wednesday, October 20th, 2021

## 

## 2. Overview and Understanding of Requirements

Ice Track is a web application in development that strives to solve shipment tracking and inventory management within Tom and Adam’s Ice Cream Company. The Ice Track software system is broken down into four key subsystems; order entry, inventory management, shipment tracking, and trouble ticket management.

The order entry subsystem helps by automating the entry of orders into the Companies order database. It will also store the data of customers so that customers can be associated with all of their orders which will allow for easier management of present and past orders from customers. The inventory management subsystem’s job is to track the ice cream inventory for Tom and Adam’s Ice Cream Company by allowing users to add, reserve, and remove products from their inventory. The shipment tracking subsystem keeps track of the status of all shipments within the database including information such as order date, number of boxes within the shipment, if the order has shipped yet, its expected delivery date, its actual delivery date, and its method of delivery. The last of the four systems is in charge of dealing with all the problems that can arise throughout the ordering, inventory management, and tracking that occurs within the system. Both reported internal production/order fulfillment/shipping problems and reported customer-facing problems are taken care of by the Trouble Ticket Management subsystem and the subsystem allows for easy resolvement of issues. These four subsystems are intertwined to deliver a software system that allows for a seamless overall management of Tom and Adam’s Ice Cream Company’s shipments and inventory.

The Ice Track software system will be able to handle all of the order, customer, and issue tracking data. It will also be able to handle multiple users simultaneously, making sure that whenever a change is made in the system all instances are updated/insync. **When performing actions on the application it will respond within 5 seconds**. The system will also be able to restore user information to its previous state if the user wants to revert previous changes.

## 

## 3. JAD Sessions/Prototypes

* **October 4th - October 10th**: Referenced the requirements database to develop screens for the login, inventory management, shipment management, trouble ticket management, and trouble ticket input screens. Each member of the Neapolitan Solutions team was tasked with designing a screen and this process was completed between Oct 4th and Oct 10th.
* **October 18th**: Referenced the requirements database when we designed the firebase database. Each team member contributed in a discussion of what the architecture of the database should be and it was based on requirements that are found within our requirements database, and we completed this JAD session during our group meeting on October 18th.

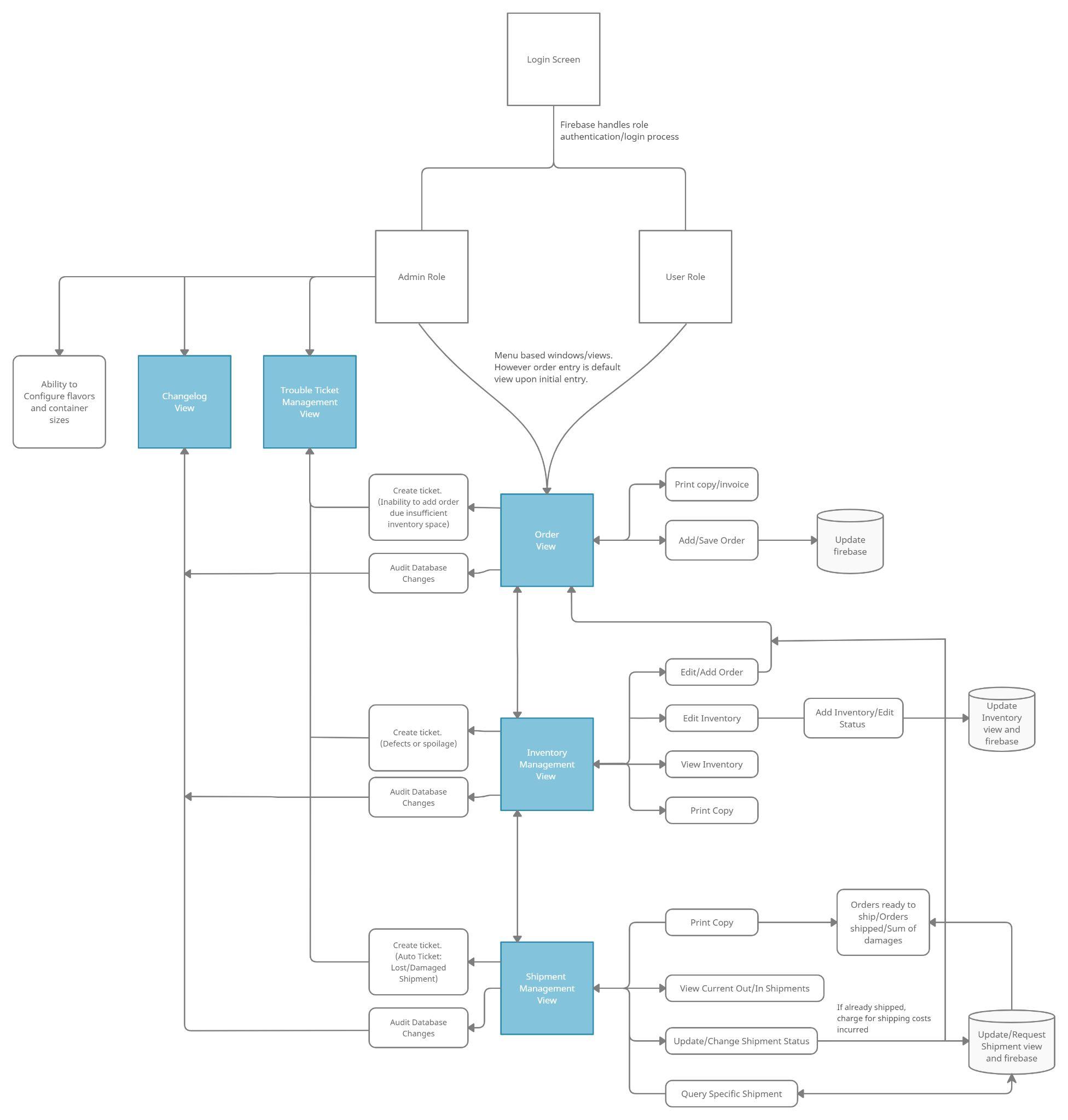
## 

## 4. Top Level Design

### Top Level Architecture Plan:

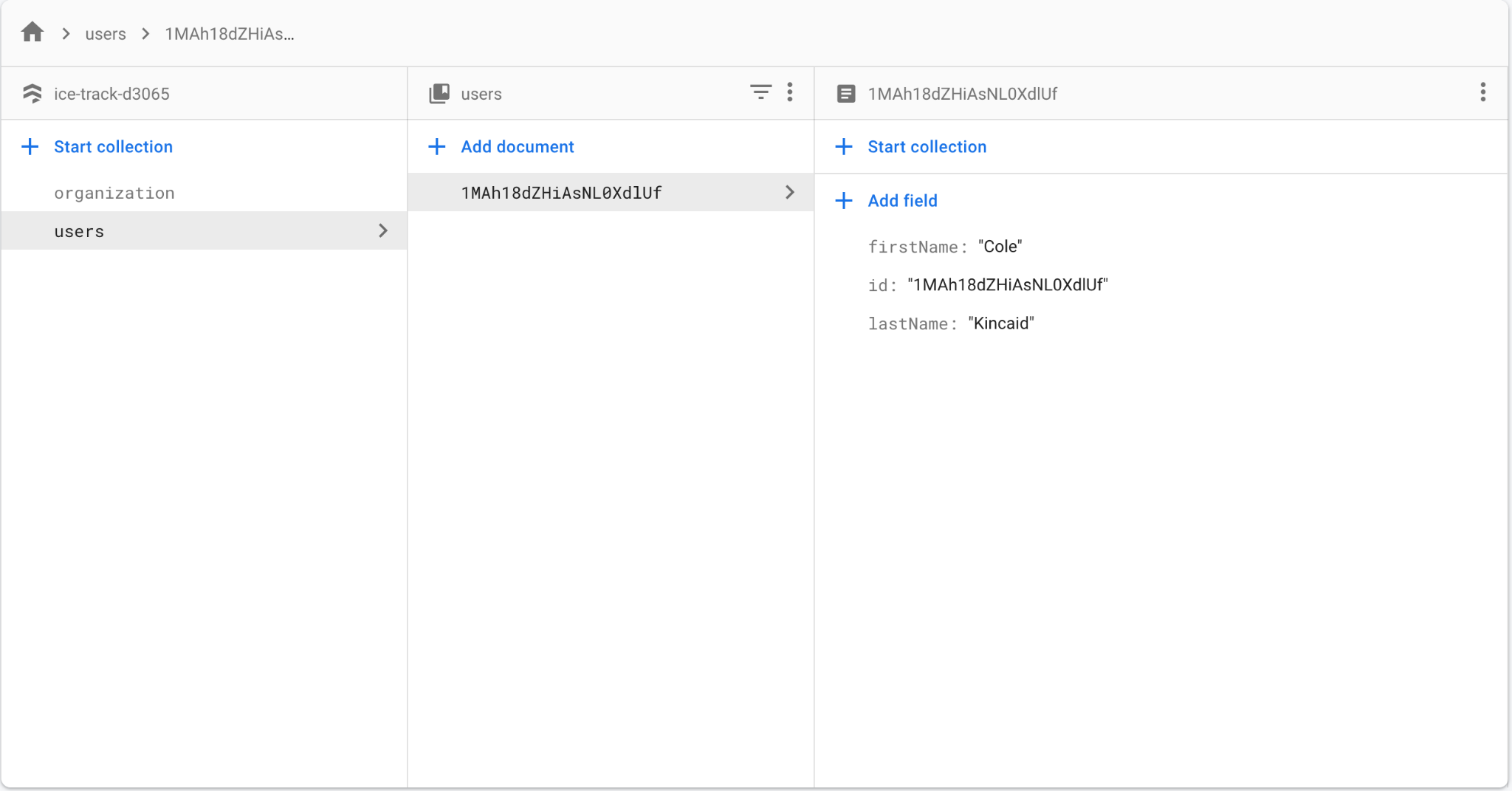
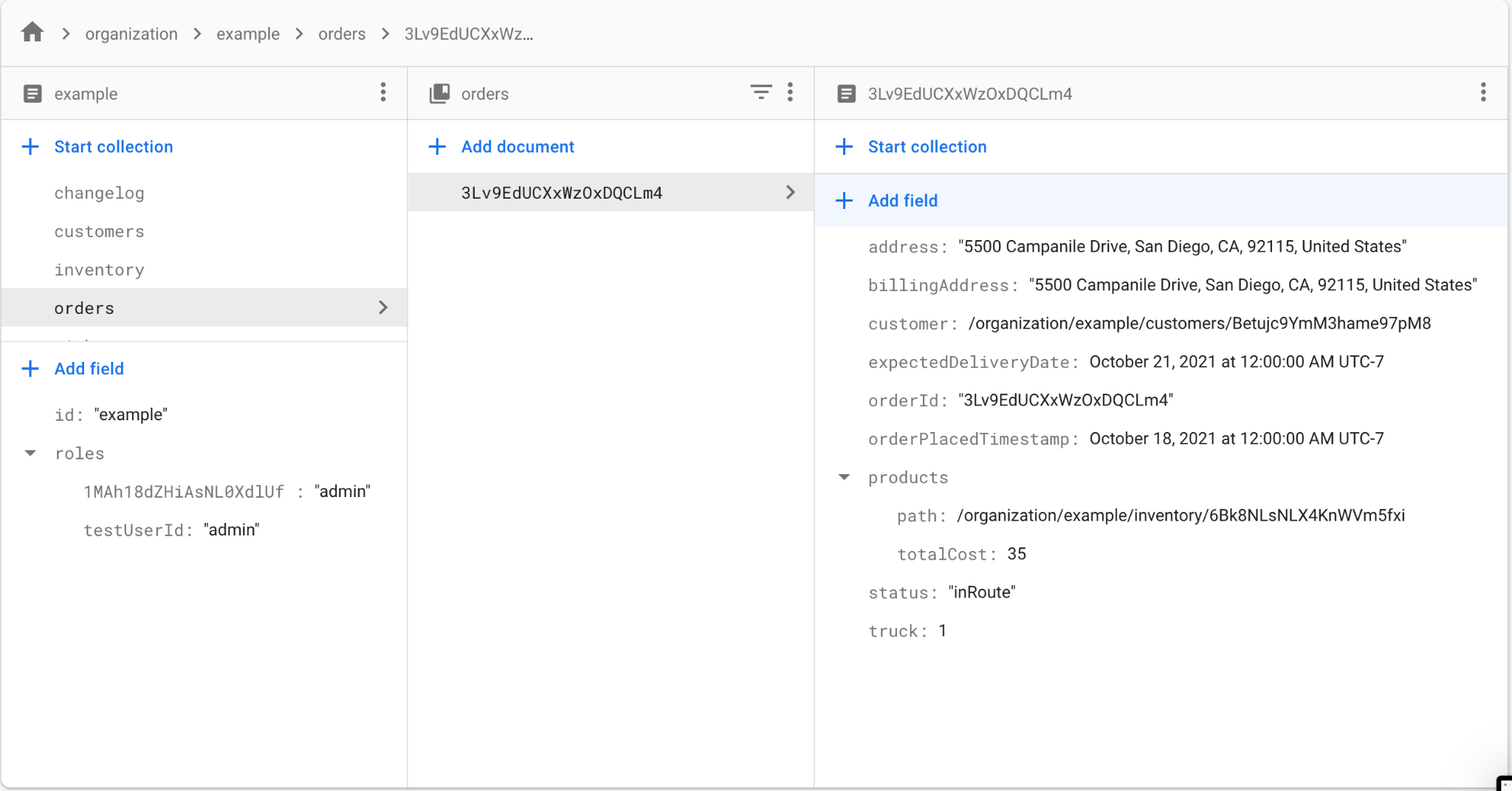
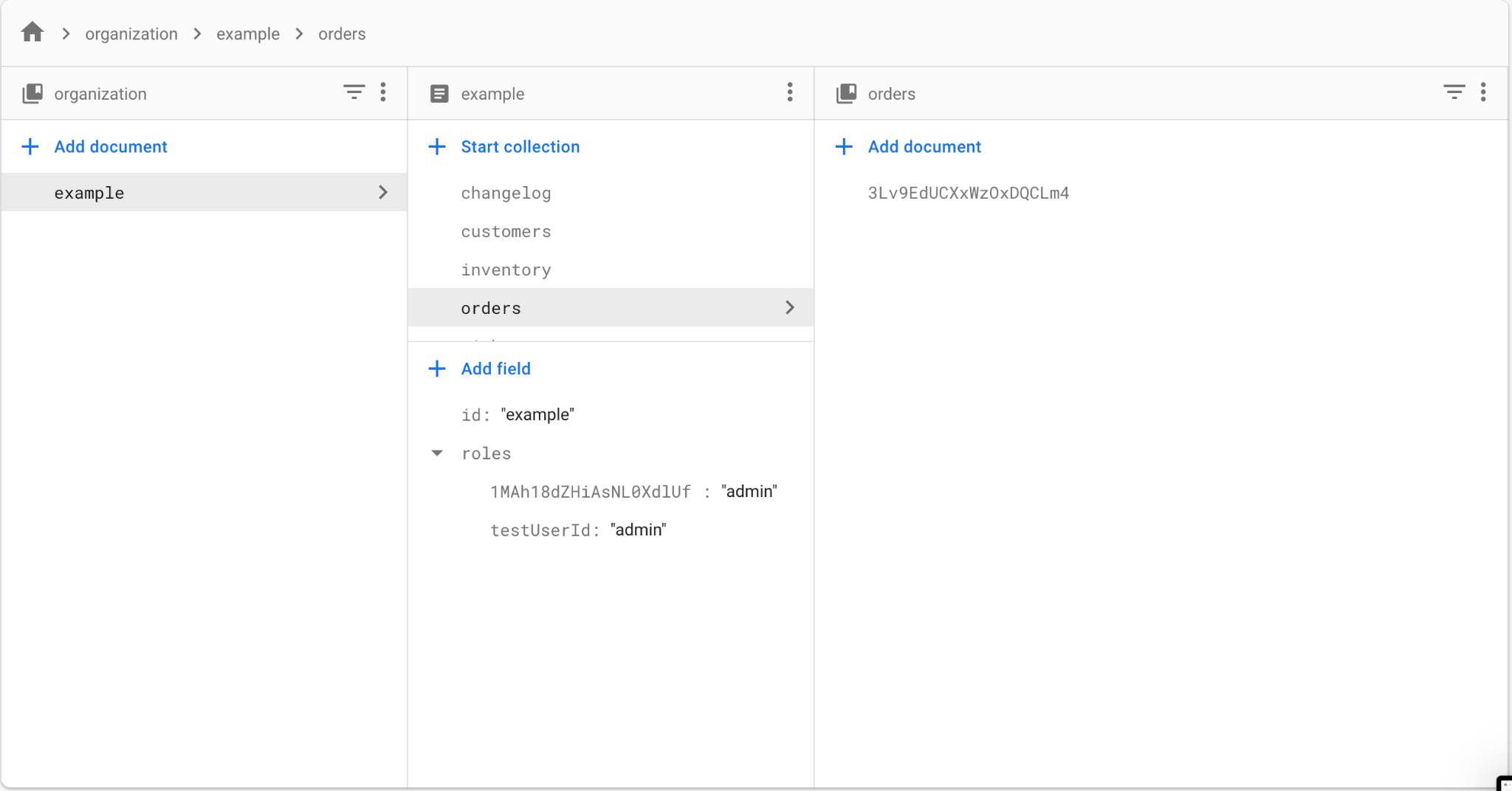
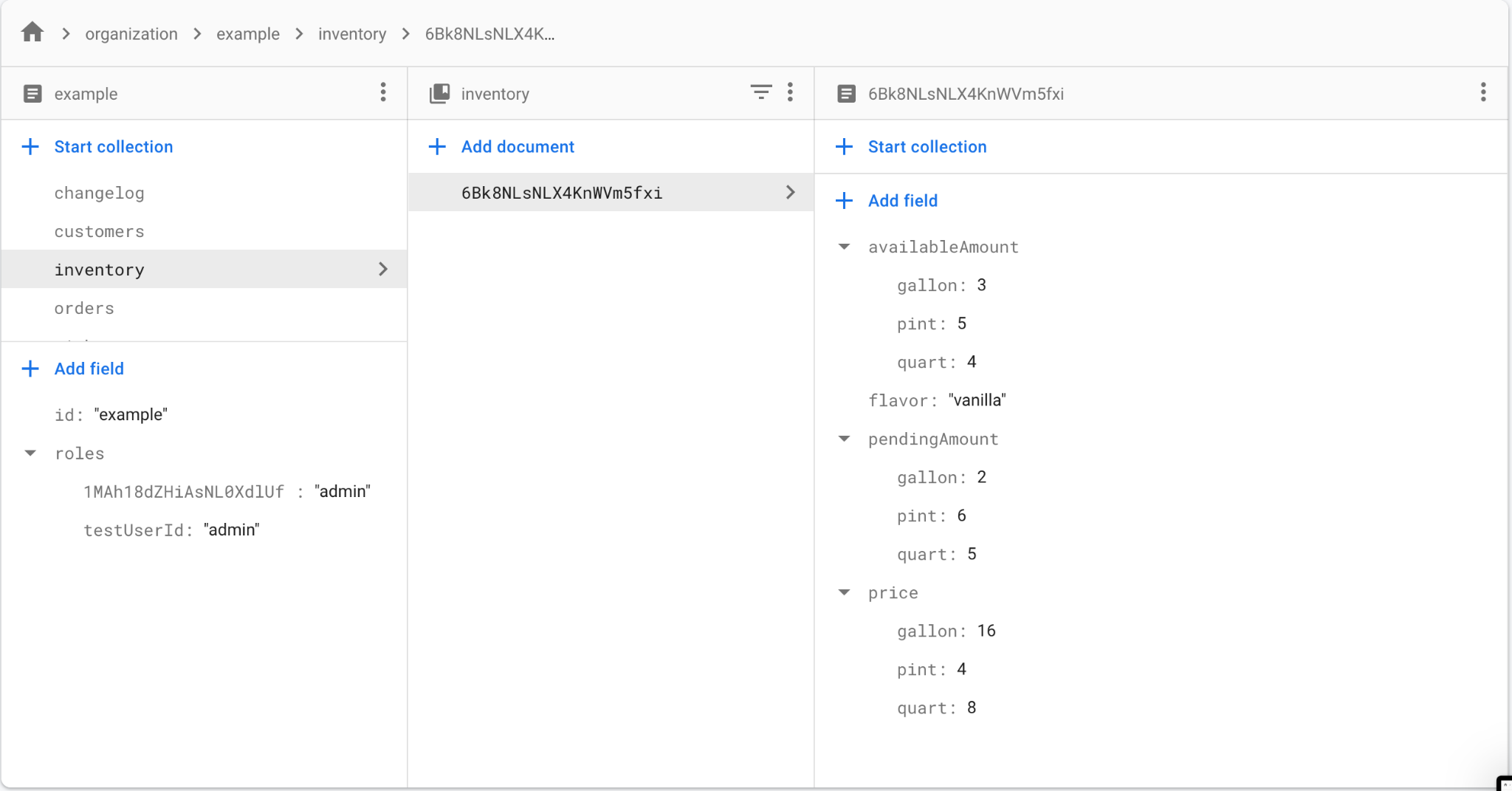
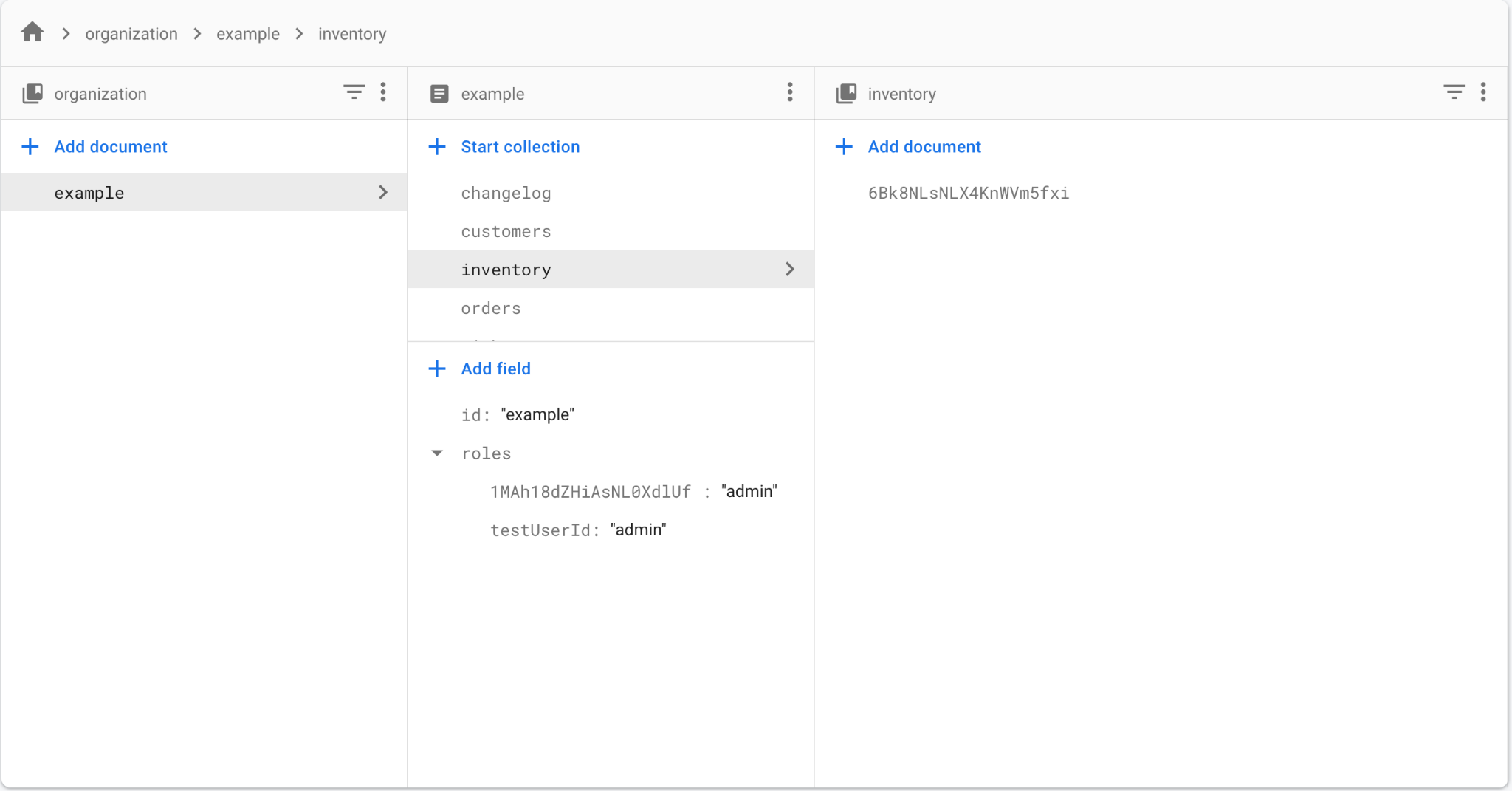
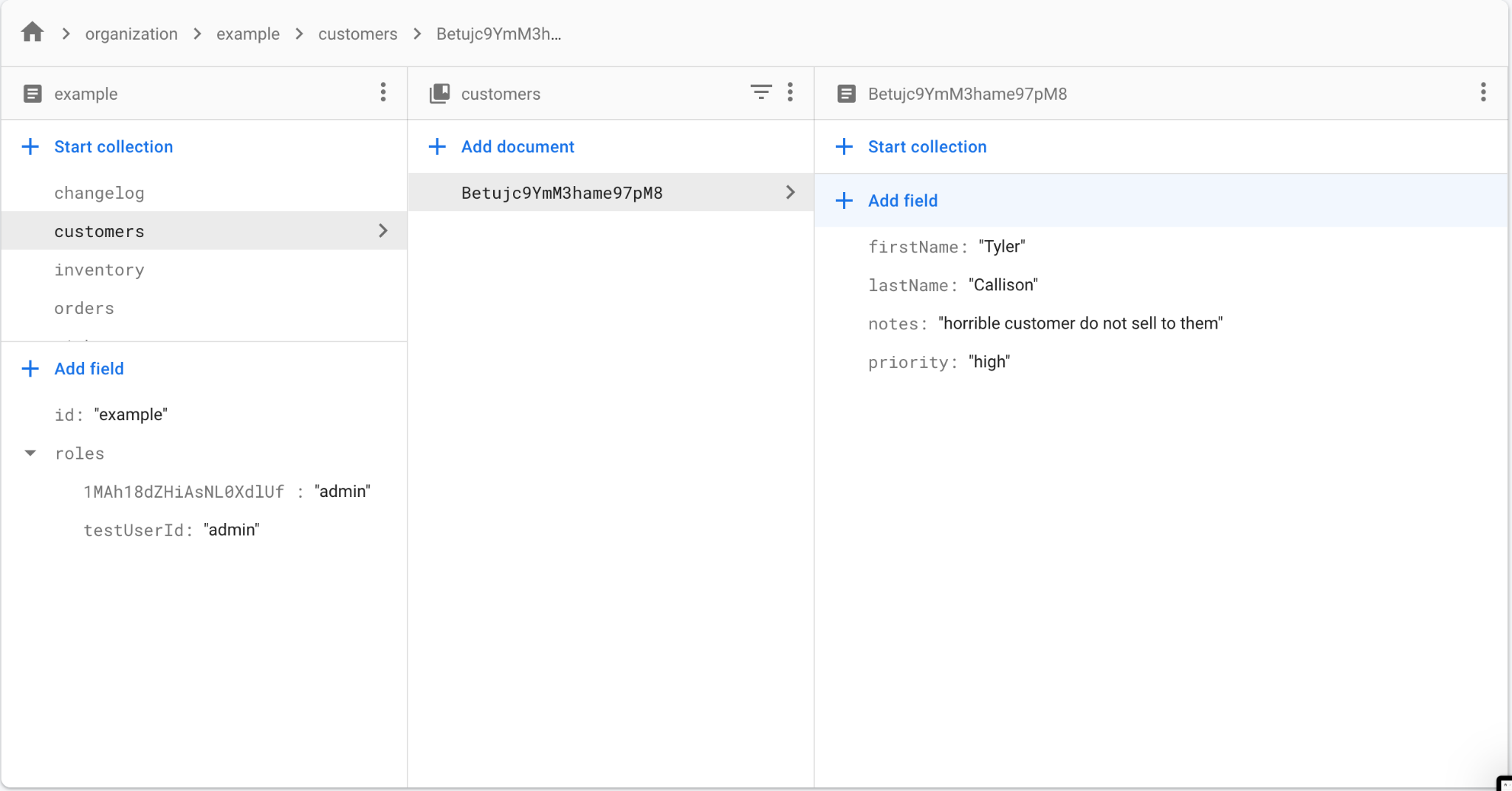
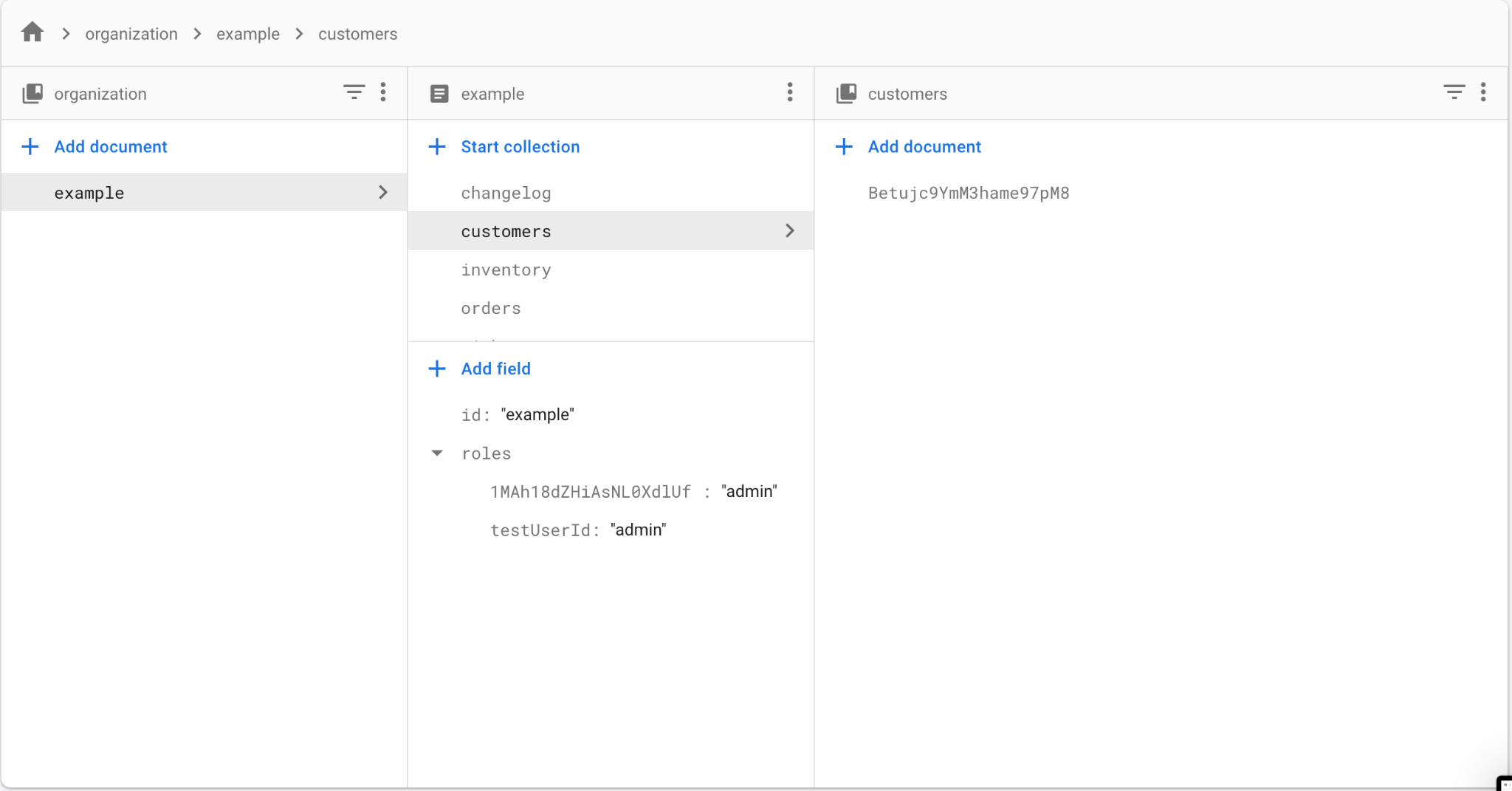
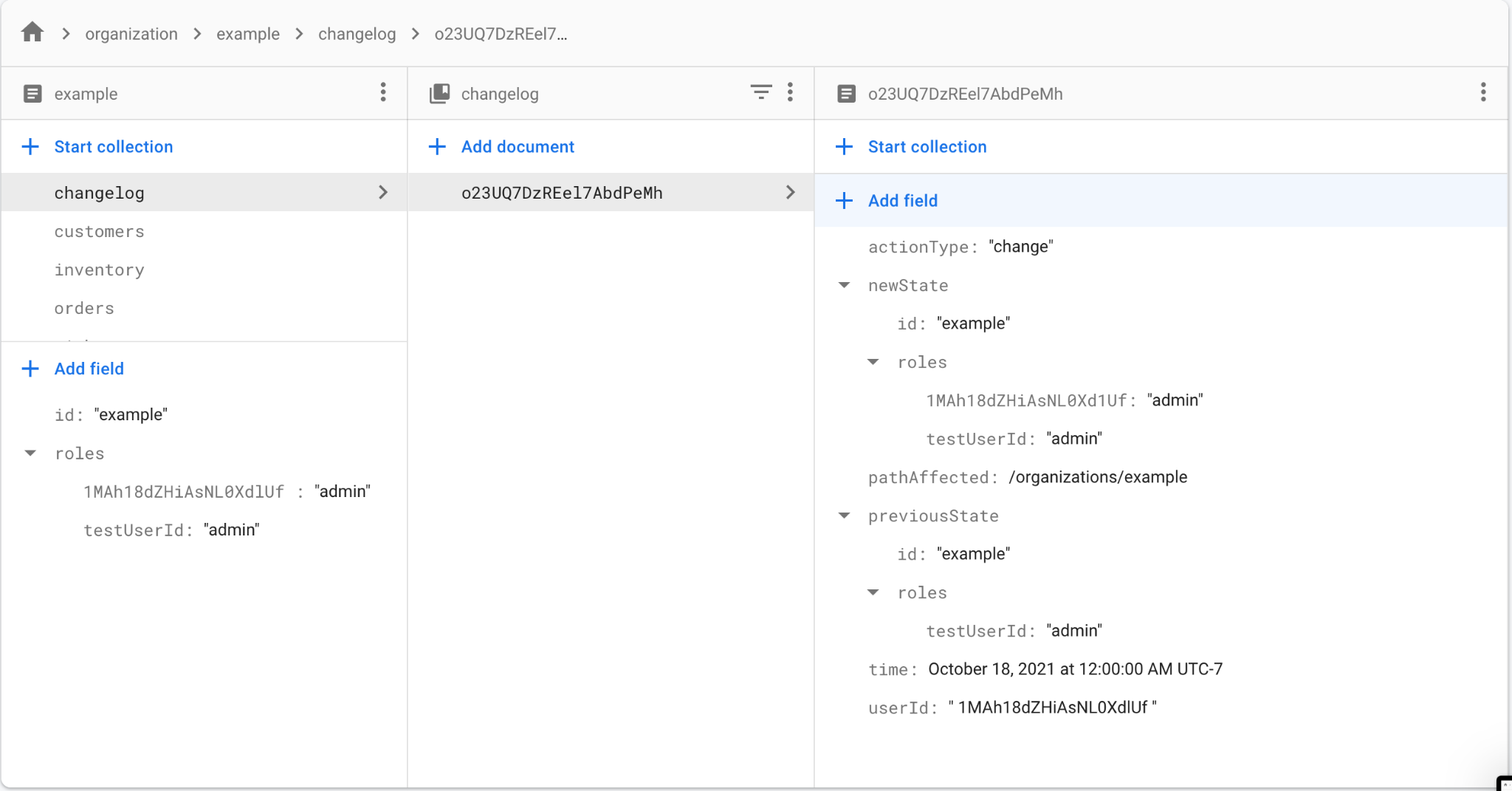
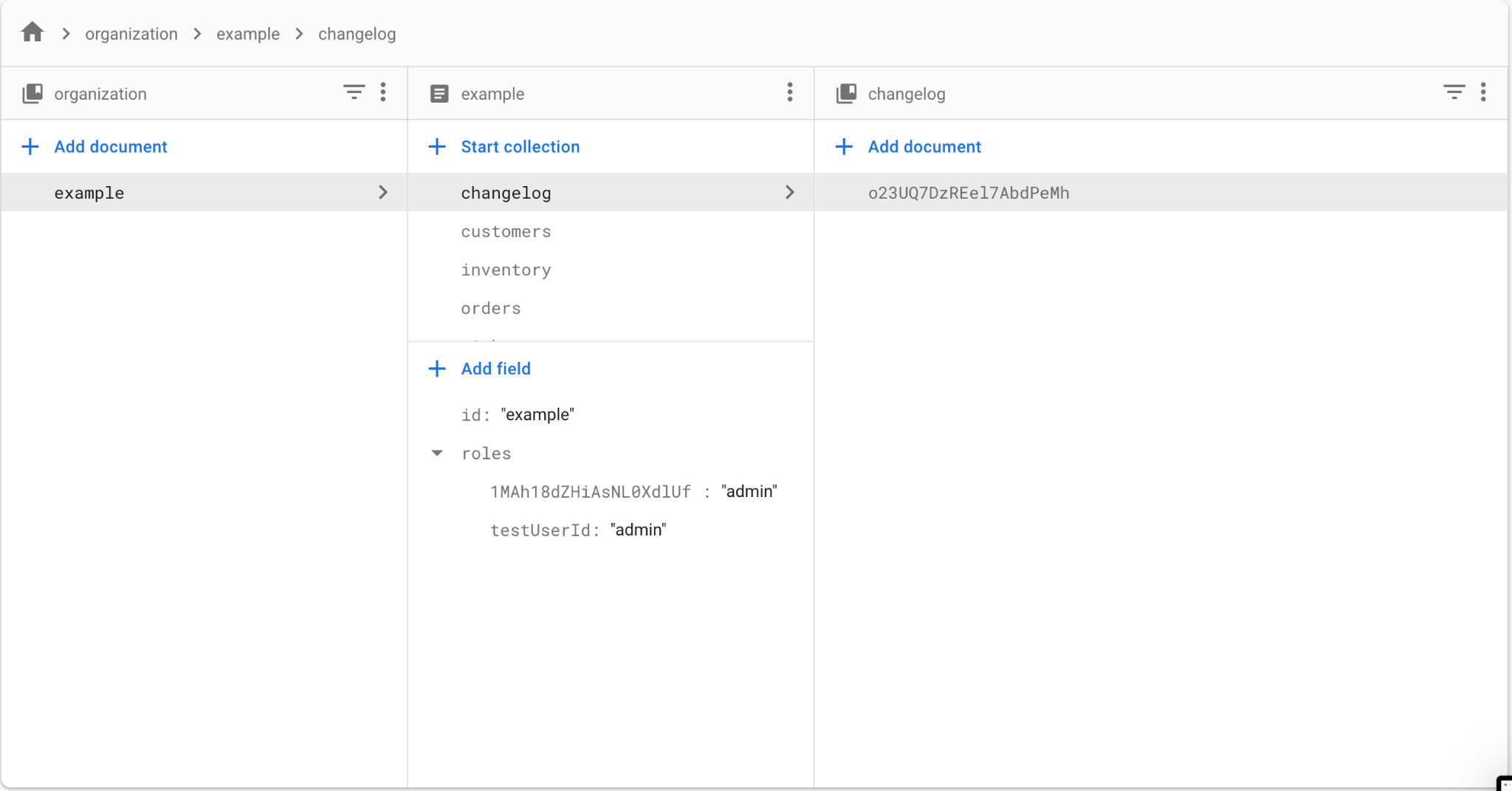
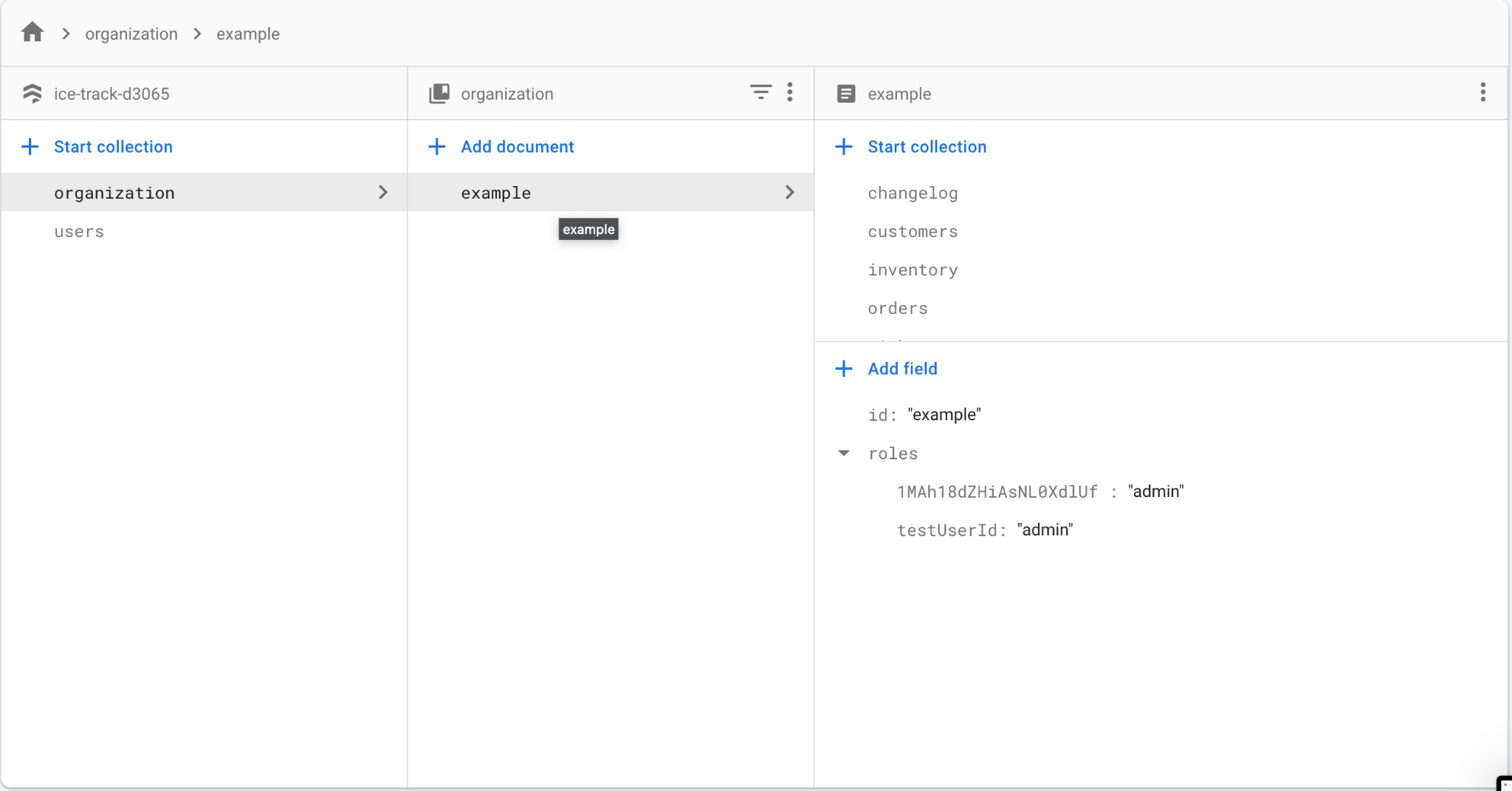
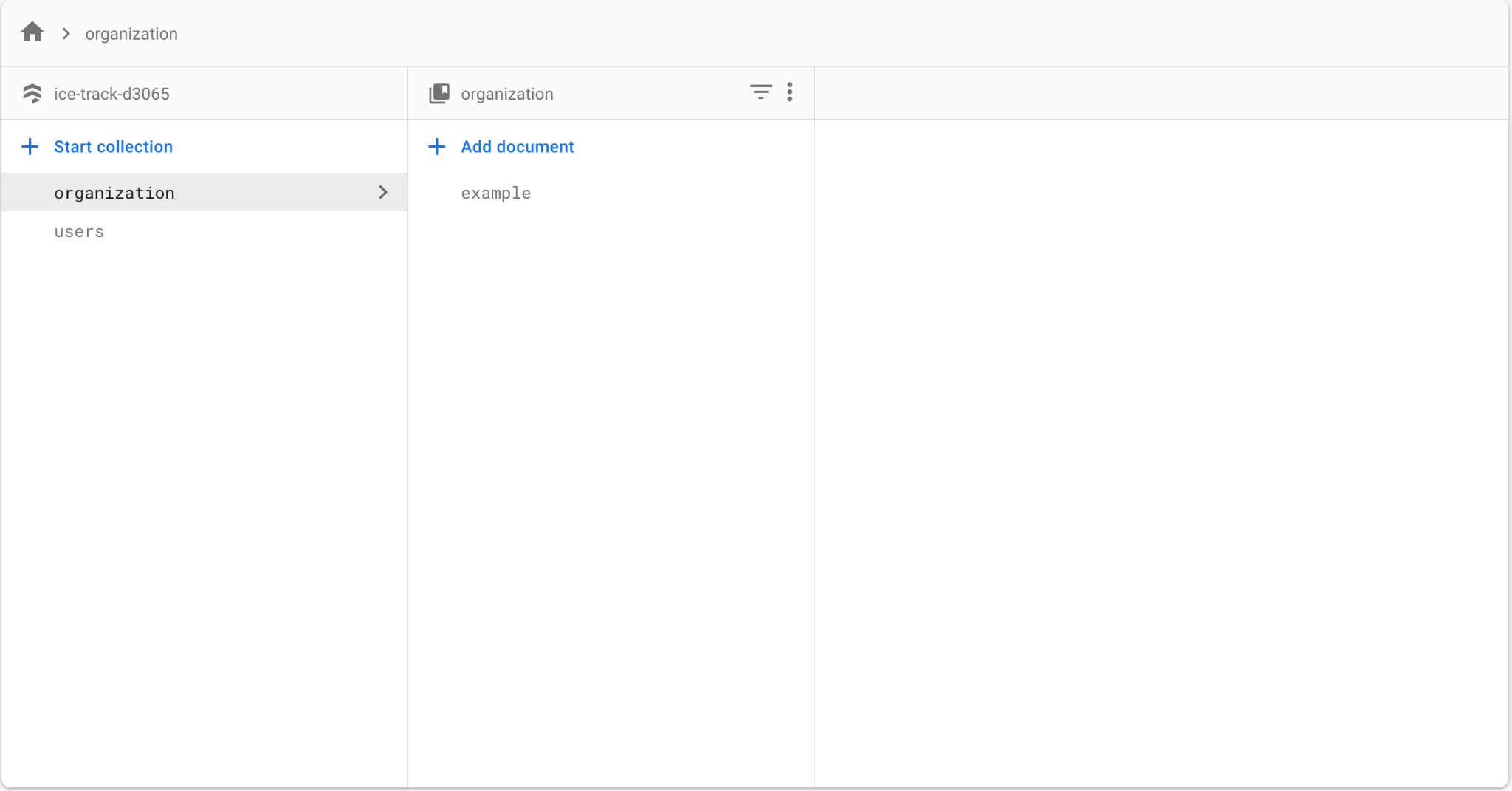
For the top level architecture diagram, we used a heterogeneous architecture style that utilizes a mixture of data abstraction, event-based

Top Level Architecture Diagram:



### Database Design:

Below are screenshots of our Firebase Firestore Database which stores changelog, customer,inventory, order, and trouble ticket data



### Requirements Mapping -

Requirements mapped to subsections of the software system are within our requirements database, which was made using Atlassian’s Confluence. [REQUIREMENTS DATABASE](https://tylercallison.atlassian.net/l/c/u31uQCuZ)

## 5. Updates to Project Plan

### 5a) Estimates:

#### Size Estimate, using Application Point Sizing:

**Using object point counting** {client} [server]

* + Inventory Management System: 1 screen, + edit modal, [1 data table]
  + Order System:
  + Customer Profiles: 1 screen + edit modal, [1 data table]
  + Orders: 3 screens + create, edit modals
    - Products available screen, [same table as inventory management]
    - Individual product info modal [also same table as inventory management]
    - Order input/Order confirmation screen (maybe utilizes both customer info and current cart data tables) + {1 data table}
    - Order/Invoice/Truck tracking screen + [1 data table]
  + Trouble Tickets:
    - 1 screen for customers to create a ticket + {1 data table}
    - 1 screen to see all tickets + modal to view each ticket individually[1 data table]
  + Changelog: 1 screen, + [1 data table]
  + TOTAL: ~7 Screens, [5 data tables] + {2 data tables}

#### Testing Software to be developed:

Our estimates for the size of the testing software to be developed are as follows:

* We will utilize lines of code as our sizing method because it is easy to compare with the functional code.
* Lines of code for testing software can often be greater than the lines of functional code, however our goal is to maintain a maximum ratio of 2:1 testing to functional lines, to ensure that we do not waste time writing and running unnecessary tests.

### 5b) Schedule and Estimate of Total Labour Hours:

[Project Schedule (Subject to Change)](https://docs.google.com/spreadsheets/d/1QeU2sPi6dW89G62PEPwvEtTfkLfFjAGA5nVIOWRuGJE/edit#gid=0)

### 5c) Defects – number detected per phase (requirements, top level design, test planning)

* Requirements (1)
  + Shipment
    - We thought the requirements wanted us to implement an incoming shipment tracking system, but the requirements only ask for a manual input of pending stock. This was handled in a JAD session with the requirements document.
* Top level design (2)
  + Database Implementation
    - Users to be general users for the entire system linked to the company or user from within the company (where the data is held). Ultimately went to the general pool so users would not have to recreate an account under strange circumstances with another company. Error prevention / handling
  + UI Layout / Design
    - Uniformity needs to be better and more consistent with a common design
* Test planning (3)
  + UI elements not always being tested yet
    - Planning on testing all UI elements with test input data. Has not been completely implemented yet for all UI elements, but is on track.
  + Data processing / transactional data
    - Not fully done, but the start of testing is occurring while development is ongoing.
  + Unit Testing
    - Automation test on build with basic inputs, outputs, and verifications.

## 6. Identification and Status of Issues/Risks (previous and current)

**Previous risks:**

Requirement analysis:

* While the chosen lifecycle model allows for some flexibility regarding the requirements. Icetrack’s core functionality must be well understood to write a consistent project notebook and code base.

**Current risks:**

Developer inexperience:

* React is a new framework for a significant portion of the development team.
* Firebase database system is a new tool for the development team.

Sprint schedule:

* Project schedule has already seen significant schedule adjustments in early stages of development.