Savvy ZooKeeper

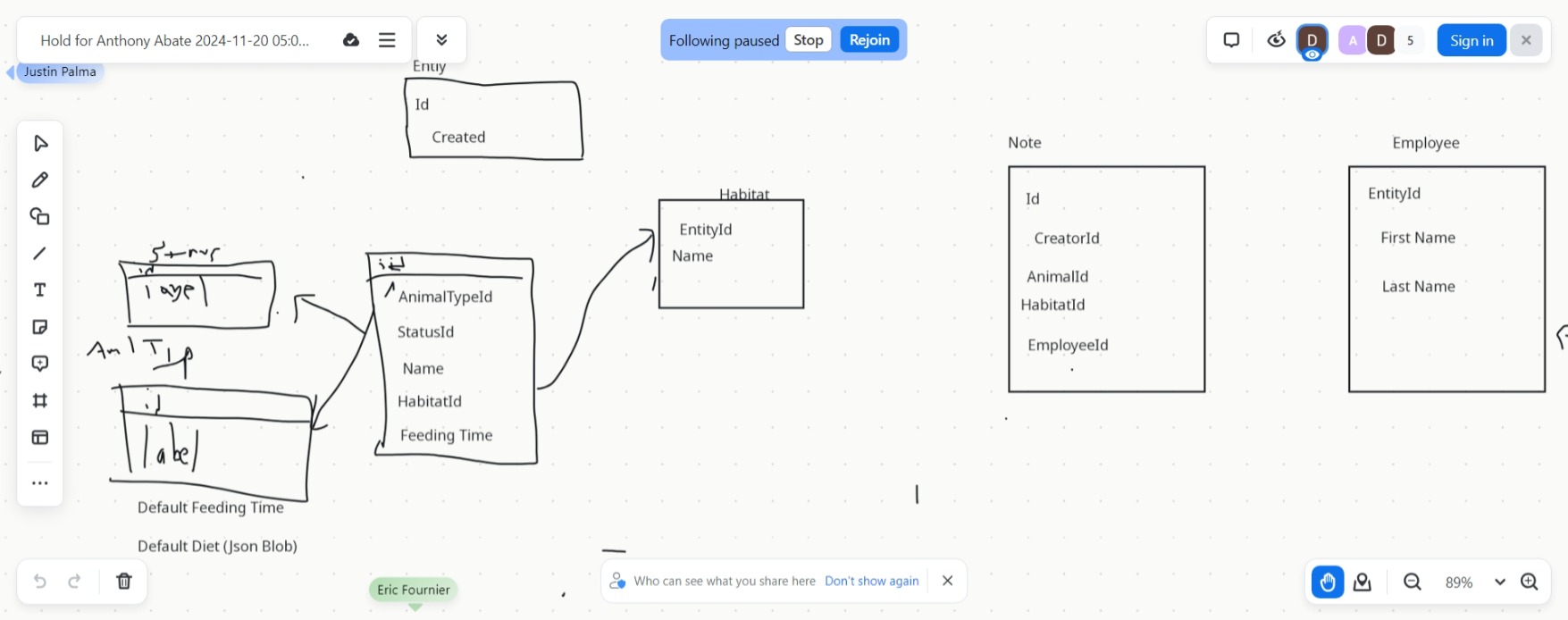
(I know there is an Apache Zookeeper, but I couldn’t think of a better name)

**Important** – This project task of ‘build an application’ is extremely broad and can. In an effort to produce a reasonable amount of ‘product’ in a reasonable amount of time there are some very important caveats

1. The data model is more or less complete with FKs, Indexes, where appropriate.
2. The REST API is a **skeleton** – except where noted, only 1 -2 endpoints have been fully implemented
3. Except to demonstrate the required APIs (ie to create animal / habitat / note) Much of the UI is **read-only**

# Data Model

First task was to come up with a data model that represented all the requirements.



The original design morphed into 3 main data schemas:

Security / Metadata / Data

## Security Tables

Security tables are all related to permissions, roles, role membership

Some take aways from the security data model:

Every record in the system needs a valid principal for tracking create and update operations.

A principal need not be associated to an employee. This is the case for the system account that creates the default data. Roles are bundles of permissions. Permissions can have an optional target entity. This means a permission of ‘read’ can be constrained to specific entity (or subtree if inherit flag is set)

## Metadata Tables

This data that is generally static supplied by the application. In our application, the Animal Types and Habitat data falls into this category. Barring unusual circumstances, species data will not change nor will customers add or be expected to maintain this dataset in application. A ‘metadata/reference data’ administrator would be responsible for these tables. Client specific overrides would be handled in other tables, but metadata should be a ‘golden’ dataset.

## Data Tables

These tables are client specific data. Employees will create and interact with records in these tables.

These tables also follow a TPT (Table-Per-Type) hierarchy with a base table **Entity**

**NoteEntity** is a many-to-many table allowing the same note to be assigned to any entity in the TPT hierarchy. Since a Note is also an Entity, Notes-Notes is actually allowed by the schema, but could be disabled via business rules or an sql check constraint. Alternatively, Notes could be split out of the TPT hierarchy.

**AnimalState** is an ‘insert only’ table containing the status of animals as a historical time series.