Business DSL

Nathan Bain
Jonathan Dobyns
Sean Wang
Rosie Wu

Goals

- Designed to easily help perform operations on a business system with simple English syntax
- Fully operational on a database, allowing for permanency
- Supports features such as meeting and project scheduling, with automatic rescheduling
- Easy to expand features

Functions

- `CREATE NEW` **`EMPLOYEE`** `WITH` **`NAME`** `AS “Morgan Freeman”`
- `UPDATE` **`PROJECT 2`** `MODIFY` **`END`** `TO “12/25/2016”`
- `ASSIGN` **`EMPLOYEE 12`** `TO EVENT` **`MEETING 5`**
- Supports Console Input, File Importing for Employees & Clients, Exporting a Calendar of Company Events

Entities In The System

- **Employees**
  - Name, Schedule, Pay, etc.
- **Clients**
  - Name, Balance, etc.
- **Meetings**
  - Client, Time, Employees
- **Projects**
  - Employees, End Date

- **Inventories**
  - Quantity, Costs, etc.
- **Purchases**
  - Client, Amounts
- **Shipments**
  - Client, Reception Date, etc

Relationships

- Entire company has a schedule of Meetings
- Each Employee has a personal Meeting schedule
- Projects are completed upon a CLOSE operation that checks the current date against the End Date

Challenges

- Learning how to use Slick
- Dealing with user-related complications
- Lots of variations
- Scala/Slick Type Restrictions

Benefits

- Allows users to easily retrieve and manage data in a specific company environment
- Allows for simple representations of company schedules and relationships
- Automatic Rescheduling for smooth changes in the company's schedule
- Easy to expand to include additional functions

Possible Expansions

- Inventory ranking reports based on sales, volume, pricing, etc.
- Add user-selection feedback that can allow for more refined features like aliasing
- Scheduling that takes into account employee performance, client priority, and past work relationships
- Location-based scheduling for on-site and off-site meetings
- Concurrency between different instances of the DSL