Requirements:

1. Авторизация через SSO по форме клюва

   Actor: User
   Command: Register / Login / Logout
   Data: User.PublicId, User.Role, User.FullName, Beak.Image
   Event: User.Created

2. Юзеру можно поменять роль

   Actor: User.Role == Admin, User.Role == Manager
   Command: SetRole
   Data: User.PublicId, User.Role
   Event: User.RoleChanged

3. Таску может создать любой человек

   Actor: User
   Command: CreateTask
   Data: User.PublicId, Task.Description
   Event: Task.Created

4. Менеджеры или администраторы должны иметь кнопку «заассайнить задачи», которая возьмёт все открытые задачи и рандомно заассайнит каждую на любого из сотрудников (Update: кроме менеджера и администратора) .

   Actor: User.Role == Admin, User.Role == Manager
   Command: AssignAllTasks
   Data: Task.PublicId, Task.Status, List<User> allUsers
   Event: Task.AssigneeChanged

5. Каждый сотрудник должен иметь возможность видеть в отдельном месте список заассайненных на него задач

   Actor: User.Role == OrdinaryPopug
   Command: GetTasksAssignedToMe
   Data: Task.PublicId, Task.Description, Task.Assignee, Task.Status
   Event: n/a

6. Сотрудник может отметить задачу выполненной

   Actor: User.Role == OrdinaryPopug
   Command: CompleteTask
   Data: Task.PublicId, Task.Assignee, Task.Status

Event: `Task.StatusChanged`

7. Детальная информация по Аккаунтингу доступна только Админам и Бухгалтерам

   Actor: `User.Role == Admin, User.Role == Accountant`
   Command: `GetAmountOfEarnedMoney`
   Data: `EarnedMoney, DateTime.Day`
   Event: `AuditLog.Action`

8. Информация о собственных счетах доступна у обычных попуг

   Actor: `User.Role == OrdinaryPopug`
   Command: `GetAccoutData`
   Data: `Account.Amount`
   Event: `ActivityLog.Action`

9. Авторизация в дешборде аккаунтинга должна выполняться через общий сервис аутентификации UberPopug Inc - SSO

   Actor: `User`
   Command: `Login`
   Data: `User.PublicId, User.Role`
   Event: `ActivityLog.Action`

10. У каждого из сотрудников должен быть свой счёт, который показывает, сколько за сегодня он получил денег. У счёта должен быть аудитлог того, за что были списаны или начислены деньги, с подробным описанием каждой из задач.

    Actor: `Task.AssigneeChanged, Task.StatusChanged`
    Command: `UpdateAccountBalance`
    Data: `Task.PublicId, Task.Price, User.PublicId, Account.PublicId, Account.Amount`
    Event: `Account.BalanceUpdated`

11. Цены на задачу определяется единоразово, в момент появления в системе (можно с минимальной задержкой)
    a. **Updated:** цены рассчитывается без привязки к сотруднику
    b. Формула, которая говорит сколько списать денег с сотрудника при ассайне задачи — `rand(-10..-20)$`
    c. Формула, которая говорит сколько начислить денег сотруднику для выполненой задачи — `rand(20..40)$`

    Actor: `Task.Created`
    Command: `SetPrice`
    Data: `Task.PublicId, Task.Price, Task.AssigneeFee`
    Event: `Task.PriceChanged (Price, AssigneeFee)`

12. Деньги списываются сразу после ассайна на сотрудника

    Actor: Task.AssigneeChanged
    Command: UpdateAccountBalance
    Data: Task.PublicId, Task.AssigneeFee, User.PublicId, Account.PublicId
    Event: Account.BalanceUpdated

13. Деньги начисляються сразу после выполения задачи

    Actor: Task.StatusChanged
    Command: UpdateAccountBalance
    Data: Task.PublicId, Task.Price, User.PublicId, Account.PublicId
    Event: Account.BalanceUpdated

14. Дешборд должен выводить количество заработанных топ-менеджментом за сегодня денег. т.е. сумма всех закрытых и заасайненых задач за день с противоположным знаком: `(sum(completed task amount) + sum(assigned task fee)) * -1`

    Actor: Task.StatusChanged, Task.AssigneeChanged
    Command: UpdateAccountBalance
    Data: Task.PublicId, Task.Price, Task.AssigneeFee, Account.PublicId
    Event: Account.BalanceUpdated

15. В конце дня необходимо:
    a. считать сколько денег сотрудник получил за рабочий день

       Actor: Scheduler
       Command: SettleAccount
       Data: User.PublicId, Account.PublicId, Account.Balance
       Event: Account.BalanceSettled

    b. отправлять на почту сумму выплаты.

       Actor: Account.BalanceSettled
       Command: SendPayment
       Data: User.PublicId, User.Email, Account.PublicId, Account.Amount
       Event: Account.PaymentSent

    c. Отрицательный баланс переносится на следующий день. Единственный способ его погасить - закрыть достаточное количество задач в течении дня.

       Actor: Scheduler

Command: SettleAccount
Data: User.PublicId, Account.PublicId, Account.Balance
Event: n/a

16. После выплаты баланса (в конце дня) он должен обнуляться, и в аудитлоге всех операций аккаунтинга должно быть отображено, что была выплачена сумма

    Actor: Account.PaymentSent
    Command: UpdateAccountBalance
    Data: Account.PublicId, Account.Balance
    Event: Account.BalanceUpdated

17. Дашборд должен выводить информацию по дням, а не за весь период сразу. (вообще хватит только за сегодня (всё равно попуги дальше не помнят), но если чувствуете, что успеете сделать аналитику за каждый день недели — будет круто)

18. Аналитика — это отдельный дашборд, доступный только админам.

    Actor: User.Role == Admin
    Command: Login / GetStatistic
    Data: User.PublicId, User.Role
    Event: n/a

19. Нужно считать, сколько заработал топ-менеджмент за сегодня и сколько попугов ушло в минус.

    Actor: User.Created, Account.BalanceUpdated
    Command: UpdateBalanceData
    Data: User.PublicId, User.Role, Account.PublicId, Account.Amount
    Event: n/a

20. Нужно указывать, сколько заработал топ-менеджмент за сегодня и сколько попугов ушло в минус.

    Actor: User.Role == Admin
    Command: GetAnalytics
    Data: TopManagmentAccount.PublicId, Account.Amount, DebtorsAmount
    Event: n/a

21. Нужно показывать самую дорогую задачу за день, неделю или месяц.

    Actor: User.Role == Admin
    Command: GetMostExpensiveTasks
    Data: Task.PublicId, Task.Price, Task.Day
    Event: n/a