Мысли:

- В системе присутствует ролевая модель и авторизация, затрагивающая все компоненты. По этой причине, кажется разумным вынести **ответственность за авторизацию пользователей и хранение данных об их аккаунтах в отдельный сервис (назовём его popug-gateway)**:
  - Плюсы:
    - **Только данный сервис взаимодействует с сервисом авторизации** popug-beak-auth-service (сервисом авторизации по клюву), далее gateway пробрасывает роль в остальные части системы с запросами (например, можно использовать кастомные хэдеры для REST)
    - **Убираем необходимость других внутренних частей системы торчать в сеть** (можно спрятать внутри условного k8s)
    - На первый взгляд, безболезненно **горизонтально масштабируется** - по своей натуре stateless, по нагрузке упирается только в возможности popug-beak-auth-serice и базу пользователей.
  - Минусы:
    - **Единая синхронная точка отказа**. Упали все реплики - упала и вся система, до остальных компонентов не достучишься
    - Есть ощущение, что будет **практически во всём заниматься boilerplate'ом, проксируя запросы в другие части системы**. Не уверен, что такой вариант с применением BFF в данном случае уместен
- В системе **выделяется контекст работы с задачами и ответственными** с вполне чёткой, на первый взгляд, изоляцией - **вынесем его в отдельный сервис popug-task-management**:
  - Полностью берёт на себя ответственность за данные по задачам и управлением им, ответственных по задачам
  - Работа с задачами осуществляется независимо от концепции биллинга или отчётности
- Отдельно выделил **сервис биллинга popug-task-billing, который управляет стоимостями задач и счетами сотрудников**, так как биллинг напрямую не пересекается с бизнес-процессом создания задач и назначения ответственных:
  - Управляет финансами каждого сотрудника в системе
  - Управляет тарификацией задач
  - Полагается на события, связанные с созданием и назначением задач
- Последний компонент системы - **сервис аналитики и отчётностей по операциям popug-task-business-analytics**. Хотя изначально сервис являлся частью биллинга, в итоге решил, что стоит вынести отдельный компонент: во-первых, формат отчётностей может поменяться по причинам, не зависимым от биллинга, во-вторых, показалось, что нагрузка по отчётностям по большей части состоит из чтений, а обновления осуществляются батчами периодически (ежедневно). Кроме того, за счёт избыточности хранения данных получаем отказоустойчивость: если ляжет биллинг, отчёты будут доступны как отдельный сервис:
  - Самая дорогая задача за день/месяц/неделю предподсчитывается каждый день на основе ограниченного количества сохраняемых данных
  - Количество сотрудников, ушедших в минус, тоже может быть предпосчитано по результатам дня
  - Лог операций за день агрегируется на основе событий по каждому сотруднику, данное действие тоже можно сделать по результатам дня
    - При этом отчёт на почту сотрудников с прибылью и операциями высылается по результатам агрегации
    - Далее панель с просмотром лога статична на протяжении дня - всегда возвращается
  - **Альтернатива при необходимости realtime отчётности**: при необходимости можно организовать real-time изменения лога событий, баланса сотрудников, самой дорогой задачи и количества ушедших в минус сотрудников путём потоковой обработки событий.
  - **Проблема**: получается, что храним баланс сотрудника в 2 местах - в биллинге и отчётности. Но, кажется, это допустимый вариант дублирования, так как мы можем менять логику подсчёта в биллинге без изменения визуализации в отчётности и наоборот.

popug-gateway:

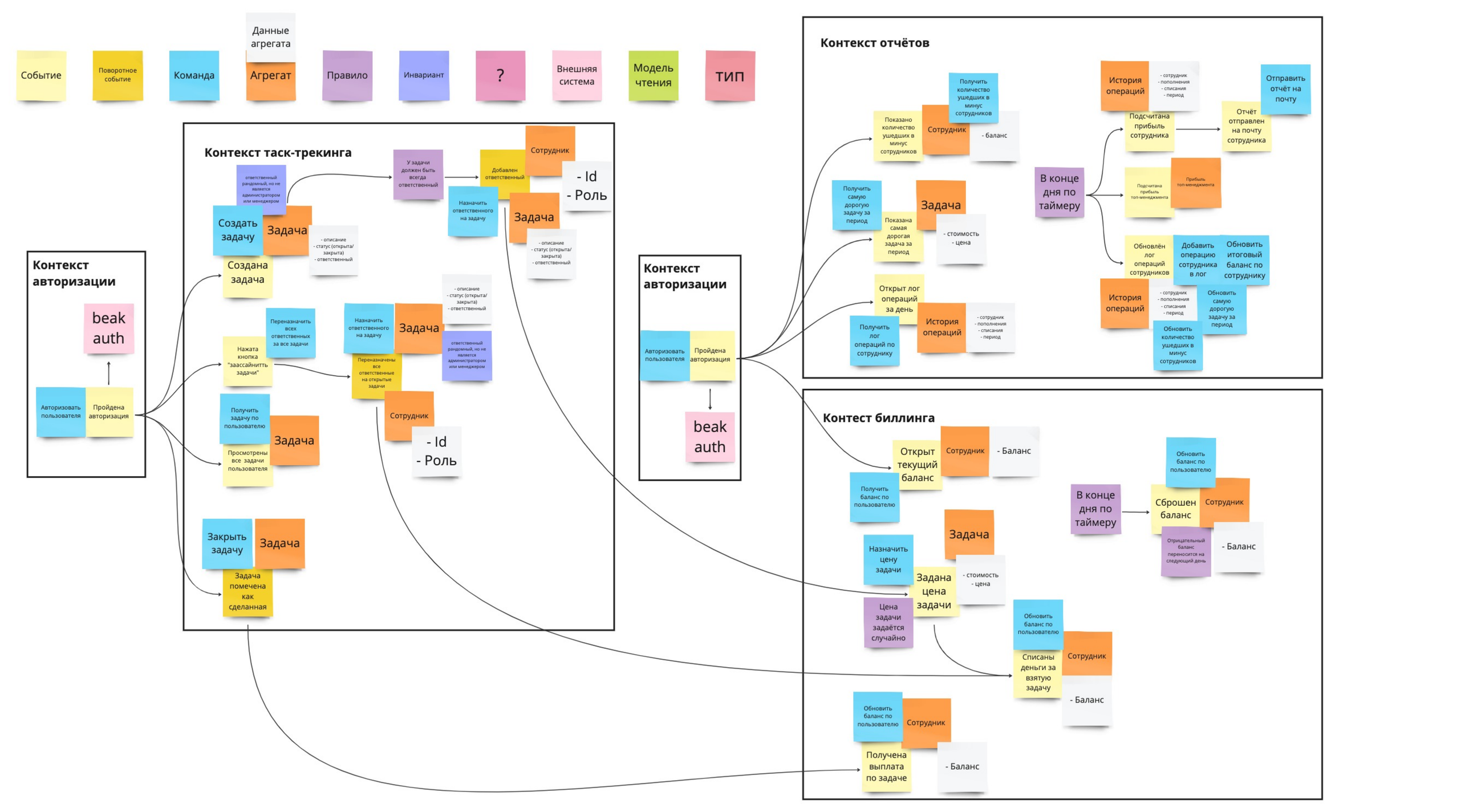
- На основе чего и как регистрируем пользователей в системе?
- Какие данные по аккаунтам попугов хранятся в системе?
- В тексте требований в списке ролей ровно 1 раз упоминается "начальник", однако дальше про эту роль ничего не говорится. Есть ли такая роль и, если да, то что она делает?
- Может ли у попуга быть несколько ролей? Если да, может ли разработчик-администратор получать задачи?
- Можно ли менять роли пользователям?

popug-task-billing:

- Стоит ли делать создание задачи с выбором случайного ответственного атомарным? На первый взгляд, кажется, что да - без ответственного у нас не должно быть самой задачи.

popug-task-billing:

- Есть ли у администраторов/менеджеров страницы лог операций и текущий баланс? Вопрос возникает в связи с неимением возможности зарабатывать этим ролям


Событие
Поворотное событие
Команда
Данные агрегата
Агрегат
Правило
Инвариант
?
Внешняя система
Модель чтения
тип
Контекст авторизации
beak auth
Авторизовать пользователя
Пройдена авторизация
Контекст таск-трекинга
ответственный рандомный, но не является администратором или менеджером
Создать задачу
Задача
Создана задача
- описание
- статус (открыта/закрыта)
- ответственный
У задачи должен быть всегда ответственный
Добавлен ответственный
Сотрудник
- Id
- Роль
Назначить ответственного на задачу
Задача
- описание
- статус (открыта/закрыта)
- ответственный
Переназначить всех ответственных за все задачи
Нажата кнопка "заассайнитть задачи"
Назначить ответственного на задачу
Задача
- описание
- статус (открыта/закрыта)
- ответственный
ответственный рандомный, но не является администратором или менеджером
Переназначены все ответственные на открытые задачи
Сотрудник
- Id
- Роль
Получить задачу по пользователю
Задача
Просмотрены все задачи пользователя
Закрыть задачу
Задача
Задача помечена как сделанная
Контекст авторизации
Авторизовать пользователя
Пройдена авторизация
beak auth
Контекст отчётов
Получить количество ушедших в минус сотрудников
Показано количество ушедших в минус сотрудников
Сотрудник
- Баланс
Получить самую дорогую задачу за период
Показана самая дорогая задача за период
Задача
- стоимость
- цена
Открыт лог операций за день
Получить лог операций по сотруднику
История операций
- сотрудник
- пополнения
- списания
- период
В конце дня по таймеру
История операций
- сотрудник
- пополнения
- списания
- период
Отправить отчёт на почту
Подсчитана прибыль сотрудника
Отчёт отправлен на почту сотрудника
Подсчитана прибыль топ-менеджмента
Прибыль топ-менеджмента
Обновлён лог операций сотрудников
Добавить операцию сотрудника в лог
Обновить итоговый баланс по сотруднику
История операций
- сотрудник
- пополнения
- списания
- период
Обновить самую дорогую задачу за период
Обновить количество ушедших в минус сотрудников
Контест биллинга
Открыт текущий баланс
Сотрудник
- Баланс
Получить баланс по пользователю
Назначить цену задачи
Задача
Задана цена задачи
- стоимость
- цена
Цена задачи задаётся случайно
В конце дня по таймеру
Обновить баланс по пользователю
Сброшен баланс
Сотрудник
Отрицательный баланс переносится на следующий день
- Баланс
Обновить баланс по пользователю
Списаны деньги за взятую задачу
Сотрудник
- Баланс
Обновить баланс по пользователю
Сотрудник
Получена выплата по задаче
- Баланс

popug-gateway:

- **popug-credentials-db**: База данных для хранения аккаунтов сотрудников и их ролей.
    - Вероятно, данные о сотрудниках (как минимум, пароли) следует хранить в зашифрованном виде
- **popug-beak-auth-service**: Сервис авторизации входящих пользовательских запросов

popug-task-management:

- **popug-task-db**: База данных для хранения задач в контексте описания и суть задач + ответственных

popug-task-billing:

- **popug-billing-bd**: База данных для хранения задач в контексте цен и стоимости + счета ответственных

popug-task-analytics:

- **popug-analytics-db**: База данных для аналитики по операциям и отчётов: лог операций по сотрудникам, динамика цен на задачи по периодам + агрегация данных - максимумы за период, количество ушедших в минус сотрудников.

Общая инфраструктура (popug-task-management, popug-task-billing, popug-task-analytics) для обмена сообщениями:

- **popug-task-events-stream**:
    - Назначение: очередь с событиями об изменениях задач и их статусов
    - События:
        - task_created: создание задачи - информация по задаче и исполнителю. **Поля**: task_id, task_description, assignee_id, timestamp
        - task_reassigned: переназначение задачи на другого ответственного. **Поля**: task_id, task_description, old_assignee_id, new_assignee_id, timestamp
        - task_completed: задача завершена указанным исполнителем. **Поля**: task_id, task_description, assignee_id, timestamp
    - Отправители: popug-task-management
    - Потребители: popug-task-billing
- **popug-billing-event-sream**:
    - Назначение: очередь с биллинговым событиями
    - События:
        - completed_task_paid: отправлен платёж по задаче исполнителю. **Поля**: task_id, task_description, assignee_id, payment_amount, timestamp
        - employee_billed_for_task: с исполнителя списаны деньги за назначение задачи. **Поля**: task_id, task_description, assignee_id, billed_sum, timestamp
    - Отправители: popug-task-billing
    - Потребители: popug-task-business-analytics

popug-gateway:

- Синхронная точка отказа системы - вопрос решается горизонтальным масштабированием;
- Отказ сервиса авторизации и/или базы пользователей: в текущем варианте - одна из главных проблем, так как при текущем дизайне система становится полностью недоступной

popug-task-management:

- Точка отказа по изменению данных по задачам - если ляжет данный сервис, текущее состояние счетов продолжит показываться и аналитика останется актуальной, однако новые задачи и изменения по существующим перестанут приходить
- Отказ базы - аналогично сервису

popug-task-billing:

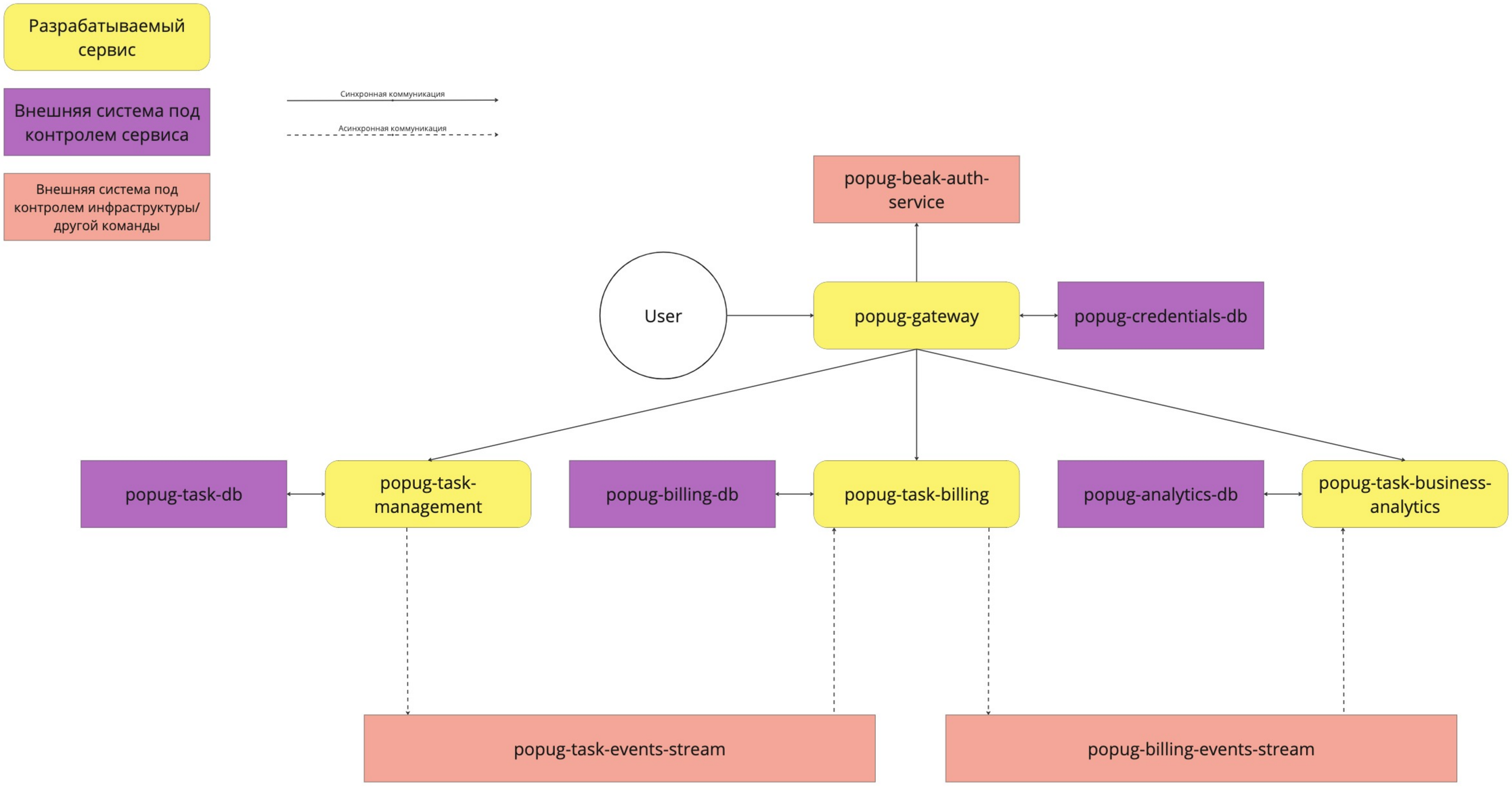
- Точка отказа по изменению состояний по счетам. В случае отказа перестанут отображаться вознаграждения по выполненным задачам, отсылаться события по платежам для аналитики. Тем не менее, ситуация не так критична, как для отказа сервиса таск-трекинга: так как сервис получает сообщения из очереди, при восстановлении работоспособности биллинга всё посчитается, только с задержкой
- Отказ базы - аналогично сервису

popug-task-analytics:

- Отказ сервиса аналитики - наименее критичная для системы ситуация, так как перестануть работать отчёты и агрегация операций. Тем не менее, при восстановлении работоспособности отчёты можно сразу пересчитать на основе данных очереди
- Отказ базы - аналогично сервису

Очереди сообщений:

- Отказ очередей сообщений - один из наименее приятных кейсов:
    - popug-task-events-stream: Если теряются сообщения по управлению задачами, то данные не дойдут до биллинга - придётся ретраить отправку событий по управлению задачами на стороне сервиса popug-task-management необходимо обеспечить переотправку информации об изменениях по задачам
    - popug-billing-event-sream: немного менее критичный кейс, чем поломка popug-task-events-stream, так как на основе событий по изменениям в задачах из popug-task-events-stream можно восстановить действия биллинга и переотправить данные по отчётам за нужный период.


Разрабатываемый сервис
Внешняя система под контролем сервиса
Внешняя система под контролем инфраструктуры/ другой команды
Синхронная коммуникация
Асинхронная коммуникация
popug-beak-auth-service
User
popug-gateway
popug-credentials-db
popug-task-db
popug-task-management
popug-billing-db
popug-task-billing
popug-analytics-db
popug-task-business-analytics
popug-task-events-stream
popug-billing-events-stream