Notification Service

Краткое описание: Читает из кафки события. В зависимости от типа события отправляет email'ы по шаблону. То как вы будете отправлять емейлы, то какой шаблонизатор вы будете использовать - полностью на ваше усмотрение.

Возможные шаблоны:

Шаблон	Описание	Микросервис
Регистрация нового пользователя.	Приветственное письмо	UserService
Восстановление пароля		UserService
Уведомление о новом появившемся рейсе.	Пользователь искал рейс, из точки А в точку Б на определенные даты. Но увы таких рейсов не было. У пользователя есть возможность подписаться на рассылку, если вдруг такой рейс появится. И если в бд появится такой рейс - пользователь будет уведомлен об этом.	TripService
Уведомление об отмене рейса		TripService
Уведомление о бронировании/ отмене бронирования билета		TicketService
Уведомление об успешной оплате/ ошибке оплаты.		FinanceService
Уведомление администратора о проблемах	Например при отправке емейла пользователю вывалился любой эксепшен, который подразумевает что письмо не дошло. В этом случае админ должен получить письмо, которое будет содержать всю необходимую информацию. (идентификатор пользователя, время, тип операции, можно даже сам эксепшен)	NotifyService

Их может быть больше, не обязательно привязываться к этим. Решите внутри команды, какие еще емейлы вам могут понадобиться.

Требования:
1) Использование шаблонов емейлов для разных типов событий. (Thymeleaf, velocity, freemaker что угодно на ваше усмотрение).
2) Шаблоны поддерживают локализацию (В зависимости от языка пользователя письма на русском/английском).
3) В конце письма хочу картинку. Нарисуйте какое-нибудь лого и вставляйте его в каждое сообщение.
4) Взаимодействие с другими микросервисами происходит через кафку.
5) Должен присутствовать rest-endpoint, который позволяет администратору отправить email в свободной форме.
6) Должен отправлять администраторам емейлы, в случае каких либо ошибок.
7) В случае ошибок при отправке писем, должен иметь какой-то механизм хранения "проблемных" емейлов. И пытаться сам повторно отправить это сообщение через определенный промежуток времени, определенное количество раз (количество попыток, и задержка между ними - должны быть настраиваемыми,

админ должен иметь возможность в рантайме менять эти значения). Если количество попыток превышено - отправить админу email, который содержит всю информацию об этой проблеме.

End-points:

№	use case	параметры
1	Как администратор я хочу иметь возможность получать статистику об отправленных писем.	**1.startDate** (optional) - выборка начиная с этой даты **2.endDate** (optional) - выборка до этой даты **3.templateType** (optional) **-** тип события, выборка емейлов по заданному типу (регистрация, смена пароля). Должен уметь принимать несколько значений. **4. email** (optional) - хочу видеть все письма, которые были отправлены на указанный адрес. 5.limit, page.
2	Как администратор, я хочу иметь возможность отправлять сообщения пользователям.	Тело post-запроса должно принимать, тему сообщения, тело и адрес.
3	Как администратор, я хочу иметь возможность повторно отправлять “проблемные” емейлы	1. **id (mandatory)**- идентификатор “проблемного письма”.

Эти минимальный список эндпоинтов, которые должны быть реализованы. Если вы считаете, что вам требуется еще какие-то - feel free to do it,

Возможные вопросы:

1) Откуда этот микросервис возьмет адреса админов?
2) Откуда этот микросервис узнает язык пользователя?
3) Откуда брать идентификатор проблемного письма?

Finance service

Сервис отвечающий за выставление счетов.

Требования:

1) Взаимодействует с NotifyService посредством кафки. После оплаты заказа, на Email пользователя должно отправляться письмо.
2) “Черная платежная коробка”. Основой workflow для этого микро сервиса выглядит следующим образом:
   1. Юзер стучится в эндпоинт для оплаты билетов. Имея id-билета и платежную информацию.
   2. Валидация платежной информации. Кроме поверхностной валидации (Отсутствие букв в cvv и cardNumber например) Должен валидироваться номер карты с помощью алгоритма Луна.
   3. Проверка ограничений на платежи по данной карте. (См. п.2 в таблице endpoints).
   4. **Rest-запрос** к черной платежной коробке.
   5. Если ЧПК “списала” деньги - меняем статус билета.
3) Обработка ошибок. Ошибки при платежах могут возникать и в ЧПК и в самом сервисе, позаботьтесь о том чтобы было очевидно, где произошла ошибка.

End-points:

	Описание	параметры
1	**Эндпоинт для оплаты билета**	1.**Id-билета** (mandatory) 2.**cardNumber**(mandatory) 3.**cvv**(mandatory) 4.**name**(optional)
2	**Блокировка/отмена блокировки платежей по определенным картам.** Администраторам надо предоставить возможность блокировать платежи по некоторым картам. Например заблокировать прием платежей с конкретной карты. Или заблокировать платежи со всех белорусских карт.	1.**cardNumber** 2.**countyCode** - один из них обязательный.
3	**Статистика по платежам.** Обычные read-операции по дате, статусу, сумме и тд.	1.**ticketId** (optional)- поиск платежки по билету. 2.**paymentStatus** (optional) - Должен уметь принимать несколько значений. 3.**startDate** (optional) 4.**endDate** (optional) 5.**amount** (optional) - сумма платежа, и оператор (больше, меньше, равно). - см. RSQL syntax. 6. page, limit

Эти минимальный список эндпоинтов, которые должны быть реализованы. Если вы считаете, что вам требуется еще какие-то - feel free to do it,

Возможные вопросы:

1. **Что такое “черная платежная коробка”?** Для эмуляции операция по банковским картам нужна какая-то песочница, которая будет эмулировать ответы платежной системы. Это может быть что угодно, главное **требование**: чтобы взаимодействие с ним происходило по ресту, и можно было прогнозировать респонс для каждой конкретной карты.
   а) **Сторонний сервис**. Многие платежные системы имеют песочницы для тестирования оплаты. Тут все также как в тестовом задании с погодой - получили ключ и стучитесь в сторонний api. Раньше для этого хорошего подходил Stripe, но сейчас они убрали эндпоинт, который принимал номер карты в сыром виде.

Возможно будет полезно:
Stripe: https://stripe.com/docs/testing
PayPal: https://developer.paypal.com/docs/payments/checkout/testing/
Square: https://developer.squareup.com/docs/testing
Разберетесь с этим - будете большими молодцами, но не усложняйте себе жизнь. Если не найдете подходящего эндпоинта, в который можно просто пихнуть номер карты и тд - не мучайтесь.
б) **Еще один микросервис**.
в) **В этом же микросервисе реализуйте какую-то заглушку**. Которая будет давать разный респонс на разные карты. - Главное чтобы вы обращались по rest к этой загулшке, а не просто дергали в сервисе.

2. Валидация номера карты? Для валидации номера карты используется алгоритм Луна, реализуете вы его сами в коде, или возьмете библиотеку, или найдете сторонний сервис - неважно, на ваше усмотрение.
3. Как определить страну. Первые 6 цифр называются "BIN" (Bank Identification Number), которые определяют банк, выпустивший карту, и страну, где была выпущена карта. - То как вы будете определять страну, тоже на ваше усмотрение. Можете найти csv, со всеми bin, и проверять в нем, можете использовать сторонний сервис.

User Service

Микросервис должен решать следующие задачи:

- регистрация;
- авторизация;
- восстановление пароля;
- выдача jwt-токена и обновление токена;
- crud операции над пользователем.

Trips microservice

Сервис реализует логику, связанную с маршрутами и рейсами.

- CRUD операции для добавления рейсов и маршрутов (Погрузиться в тему и продумать обязательные, опциональные поля для описания рейса: номер рейса, модель самолета, включен завтрак и тп). При создании рейса должно происходить создание билетов разного класса (продумать как это реализовать логично например зависимости от модели самолета есть дефолтное количество билетов и возможность задавать другие данные)
- Поиск рейсов по заданному маршруту, дате(обязательные поля) и расширенный поиск по количеству пассажиров, классу, выбрать обратный рейс. Возможность посмотреть самый быстрый, самый дешевый
- Подписка на информацию по рейсу (появление свободных билетов изменение цены и тп)
- Просмотр информации по рейсу (свободные, забронированные, купленные билеты), какая ручная кладь, включен\не включен завтрак

Tickets microservice

- Бронирование билета с дедлайном оплаты
- Отмена бронирования
- Возможность оплатить доп багаж, заказать несколько билетов
- Оплата билета (заказа)
- Просмотр истории заказов
- Предоставление скидки в 10% (но не более 50$) если день полета приходится на др

API-gateway

Является единой точкой входа в приложение. Перехватывает все запросы к приложению, и маппит не красивые длинные эндпоинты вида “/usersevice/v1/authorize” к красивыми и коротким “/auth”.
Обязательно использование сваггера.