Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Backstage Developer Portal: единая точка входа разработки

Разворачиваем Backstage 1.53: каталог сервисов, scaffolder-шаблоны, TechDocs, SSO и Kubernetes

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

У клиентов с командами разработки 10–40 человек знания о сервисах живут в головах и чатах: онбординг тянется неделями, новый репозиторий заводят вручную, документация разбросана. Мы разворачиваем Backstage как внутренний портал разработчика: каталог сервисов с владельцами, шаблоны проектов и docs-as-code. Без этого растут сроки фич и зависимость от «единственного, кто знает».

Почему это важно бизнесу

- Онбординг разработчика сокращается с 2–3 недель до дней: весь ландшафт сервисов, владельцы и доки — в одном каталоге
- Новый сервис по шаблону за 15 минут вместо 2–3 дней ручной настройки репозитория, CI и прав доступа
- Уход ключевого инженера перестаёт быть катастрофой: ownership и документация зафиксированы в catalog-info.yaml
- Меньше заявок в DevOps/админам: типовые операции разработчики выполняют сами через self-service
- Стандарты стека соблюдаются автоматически: путь по шаблону короче, чем «сделать по-своему»

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

1.53

Ставим этот main-line релиз; обновление — ежемесячно, через yarn backstage-cli versions:bump

backstage.io, релиз 07.2026

30 мин

schedule.frequency провайдеров каталога github/githubOrg; timeout 15 мин, чтобы циклы синка не...

наш стандарт

PG 16

PostgreSQL под каталог и состояние плагинов (client: pg); SQLite оставляем только для локально...

по докам backstage.io

Node 22

LTS-рантайм backend (Backstage 1.53 поддерживает 22 || 24); фиксируем в engines и в образе nod...

наш стандарт

v1beta3

apiVersion scaffolder-шаблонов (scaffolder.backstage.io/v1beta3) — на нём все наши skeleton-ре...

по докам Backstage Scaffolder

x2

Минимум реплик backend за ingress: перезапуск узла Kubernetes не роняет портал

наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Ядро портала и каталог сервисов

Что настраиваем

Backstage frontend+backend в Kubernetes клиента, PostgreSQL отдельным инстансом вне кластера

Как мы это делаем

1. Собираем приложение из @backstage/create-app; конфиг разносим на app-config.yaml и app-config.production.yaml, секреты — только ${ENV}-подстановки
2. Деплой Helm-чартом backstage/backstage: 2 реплики backend, ingress с TLS, база — PostgreSQL 16 вне кластера (backend.database.client: pg)
3. Каталог наполняем discovery-провайдером catalog.providers.github (ищет catalog-info.yaml по всей организации): schedule.frequency 30m, timeout 15m; интеграция через…
4. Оргструктуру (User/Group) тянем провайдером catalog.providers.githubOrg; в catalog.rules явно разрешаем Component, API, Resource, System, Template, Location, User,…
5. В каждый репозиторий кладём catalog-info.yaml с owner, system, lifecycle — сервис без владельца в каталог не принимаем

РЕЗУЛЬТАТ

Через 2–3 недели у клиента полный каталог: каждый сервис имеет владельца, статус жизненного цикла и связи с API и инфраструктурой. Вопросы «кто отвечает за этот сервис» и «где его репозиторий» перестают ходить по чатам — это первая метрика, которую мы снимаем после запуска.

КЛЮЧЕВОЙ НЮАНС

GitHub App вместо личного токена обязателен: у PAT общий rate limit и владелец-человек. На организациях от 100 репозиториев дополнительно тюним pageSizes (teams, teamMembers, organizationMembers) у githubOrg-п…

РЕАЛИЗАЦИЯ 2 ИЗ 3

Self-service: scaffolder-шаблоны и TechDocs

Что настраиваем

Шаблоны новых сервисов и docs-as-code под типовые стеки клиента: backend-сервис, фронт, интеграция

Как мы это делаем

1. Пишем Template (apiVersion scaffolder.backstage.io/v1beta3): parameters с ui:field OwnerPicker, валидация имени pattern '^[a-z0-9-]+$'
2. Steps шаблона: fetch:template из skeleton → publish:github с protectDefaultBranch → catalog:register нового компонента
3. В skeleton зашиваем CI-workflow, Dockerfile, catalog-info.yaml и mkdocs.yml — новый сервис сразу соответствует стандарту компании
4. TechDocs по прод-схеме: techdocs.builder: 'external', в CI-пайплайне techdocs-cli generate + techdocs-cli publish в S3-совместимое хранилище (publisher.type: awsS3,...

РЕЗУЛЬТАТ

Новый сервис заводится за 10–15 минут силами самого разработчика: репозиторий, CI, документация и регистрация в каталоге — одним прогоном шаблона. Стандарты соблюдаются не приказом, а тем, что путь по шаблону — самый короткий и удобный.

КЛЮЧЕВОЙ НЮАНС

techdocs.builder: 'local' в production оставлять нельзя: генерация mkdocs грузит backend и требует Python в его контейнере. Сборку выносим в CI, backend только читает готовую статику из хранилища с read-only д...

РЕАЛИЗАЦИЯ 3 ИЗ 3

Доступ, Kubernetes-плагин и эксплуатация

Что настраиваем

Аутентификация, permission framework и наблюдаемость портала на площадке клиента

Как мы это делаем

1. Вход через GitHub OAuth или Keycloak (OIDC): auth.environment: production, sign-in resolver пускает только пользователей, существующих в каталоге
2. Включаем permission.enabled: true и политику: удаление сущностей каталога и служебные шаблоны — только группе platform-team
3. Kubernetes-плагин: serviceLocatorMethod.type: multiTenant, кластеры в clusterLocatorMethods (type: config), ServiceAccount строго read-only
4. Эксплуатация: probes на /.backstage/health/v1/liveness и /readiness, метрики в Prometheus, алерт на ошибки синка каталога, ежедневный pg_dump базы каталога

РЕЗУЛЬТАТ

Разработчик видит поды, деплойменты и ошибки своего сервиса прямо в карточке компонента по аннотации backstage.io/kubernetes-id — без kubectl и заявок админам. Портал при этом закрыт: гостевого входа нет, разрушающие операции ограничены политикой.

КЛЮЧЕВОЙ НЮАНС

Без sign-in resolver'а с проверкой по каталогу авторизоваться сможет любой аккаунт GitHub. Это проверяем первым: resolver обязан выдавать identity только для User-сущностей организации клиента.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **SQLite остаётся в production**
  better-sqlite3 из create-app теряет каталог при пересоздании контейнера; сразу переводим на PostgreSQL — client: pg в production-конфиге

- ✕ **Гостевой вход не отключён**
  auth.providers.guest удобен на dev и опасен в бою: убираем его из production-конфига и выставляем auth.environment: production

- ✕ **Секреты в app-config уходят в git**
  Пароль БД и clientSecret держим только как ${POSTGRES_PASSWORD} и т.п.; значения — из Kubernetes Secret, конфиг в репозитории чистый

- ✕ **PAT вместо GitHub App**
  Личный токен упирается в rate limit и «умирает» вместе с учёткой сотрудника; GitHub App даёт выше лимиты и не привязан к человеку

- ✕ **Каталог наполняют вручную**
  Регистрация сервисов по одному URL не масштабируется: ставим github/githubOrg-провайдеры с schedule — каталог сходится сам

- ✕ **Релизы копятся год**
  Backstage выходит ежемесячно; пропуск 10+ версий превращает апгрейд в проект. Держим цикл: yarn backstage-cli versions:bump раз в месяц

- ✕ **TechDocs собираются на backend**
  builder: 'local' тянет Python/mkdocs в контейнер backend и вешает его на больших доках; сборку выносим в CI, publisher — awsS3

- ✕ **Портал без владельца-процесса**
  Backstage — продукт, а не разовая инсталляция: без platform-роли каталог протухает. Закрепляем владельца и SLA на актуальность карточек

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Backstage на Docker Compose, PostgreSQL 16, вход через GitHub OAuth
- Каталог через github/githubOrg-провайдеры, catalog-info.yaml в каждом репозитории
- Секреты только через переменные окружения, гостевой вход отключён

НОРМАЛЬНО

- Деплой в Kubernetes Helm-чартом: 2 реплики backend, внешняя PostgreSQL
- 3–5 scaffolder-шаблонов под типовые сервисы + TechDocs со сборкой в CI
- Ежемесячный versions:bump и smoke-тест портала после каждого апгрейда

ХОРОШО

- SSO через Keycloak/OIDC, permission framework с политиками по группам
- Kubernetes-плагин с read-only ServiceAccount в каждом кластере
- Кастомные scaffolder-actions под внутренние системы: CI, DNS, секреты
- Метрики Prometheus, алерты на ошибки синка, ежедневный pg_dump каталога

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Каталог хранится в PostgreSQL, а не в SQLite из коробки?
- [ ] Гостевой вход отключён, а sign-in resolver пускает только пользователей из каталога?
- [ ] У каждого сервиса в catalog-info.yaml заполнены owner и lifecycle?
- [ ] Каталог синхронизируется провайдерами по расписанию, а не ручной регистрацией URL?
- [ ] Интеграция с GitHub идёт через GitHub App, а не через личный токен сотрудника?
- [ ] TechDocs собираются в CI-пайплайне, а не на backend портала?
- [ ] Новый сервис заводится шаблоном за минуты — вместе с CI и регистрацией в каталоге?
- [ ] Backstage обновляется ежемесячным циклом, а не раз в год?
- [ ] Есть ежедневный бэкап базы каталога и алерт на ошибки синка?
- [ ] Права на разрушающие операции ограничены permission-политикой?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Разворачиваем Backstage под ключ: от пилота на Docker Compose до Helm-деплоя в Kubernetes с внешней PostgreSQL
- Наполняем каталог: провайдеры GitHub/GitLab, catalog-info.yaml по всем репозиториям, оргструктура и ownership
- Пишем scaffolder-шаблоны и кастомные actions под ваш стек и CI/CD
- Настраиваем SSO (Keycloak/OIDC), permission framework и TechDocs со сборкой в CI
- Сопровождаем портал: ежемесячные апгрейды, мониторинг, бэкапы, развитие плагинов

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 Backstage Docs: Deploying Backstage (Kubernetes, Docker) (backstage.io — 1.53)

02 Catalog: GitHub Discovery & GitHub Org Entity Provider (backstage.io — 1.53)

03 TechDocs Architecture & Publishers (backstage.io — 1.53)

04 Auth & Identity: Sign-in Resolvers (backstage.io — 1.53)

05 Permission Framework (backstage.io — 1.53)

06 Backend System: Root Health Service (backstage.io — 1.53)

07 Backstage Helm Charts (github.com/backstage/charts — 2026)

08 Шаблон ITfresh: app-config.production.yaml + skeleton-репозитории (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.