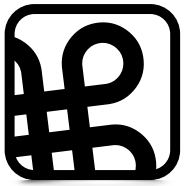


Macoun

Schon mal an CloudKit gedacht?

Friedrich Ruynat (@hdrxs)

Beispiel: Rezeptbuch-App

- **Sync:** Mac und iOS-App
- Sharing zwischen Nutzern
- **Community:** Öffentliche Rezepte
- Hobby? Indie-Entwickler?

Eigenes Backend?

- ⊖ **Keine Expertise:**
  Technologien? Anbieter?
- ⊖ **Ruhig Schlafen:**
  Sicherheit, Verfügbarkeit, Rechtliche Aspekte, Kosten?
- ⊖ **Eigene Logins:**
  Einstiegshürde, Nutzervertrauen, Datenschutz…

Alternative: CloudKit

- **Apples Backend-as-a-Service:**
  Datenbank, Push Notifications, Web-API, Suche, ...
- Wird nicht so schnell verschwinden
- Schützt Nutzerdaten vor Entwicklern
- (Keine) Accounts

Kosten(los)!

- Apple: Missbrauch vermeiden
- **Private Daten:** iCloud-Quota des Nutzers
- **Öffentliche Daten:** Kostenloses Inklusiv-Volumen pro App

Bis 4 Mio

	Dateien	DB	Rx
pro User	250 MB	2,5 MB	50 MB
Gesamt	10 TB	1 TB	200 TB

Kosten(los)!

- Apple: Missbrauch vermeiden
- **Private Daten:** iCloud-Quota des Nutzers
- **Öffentliche Daten:** Kostenloses Inklusiv-Volumen pro App

4 - 10 Mio

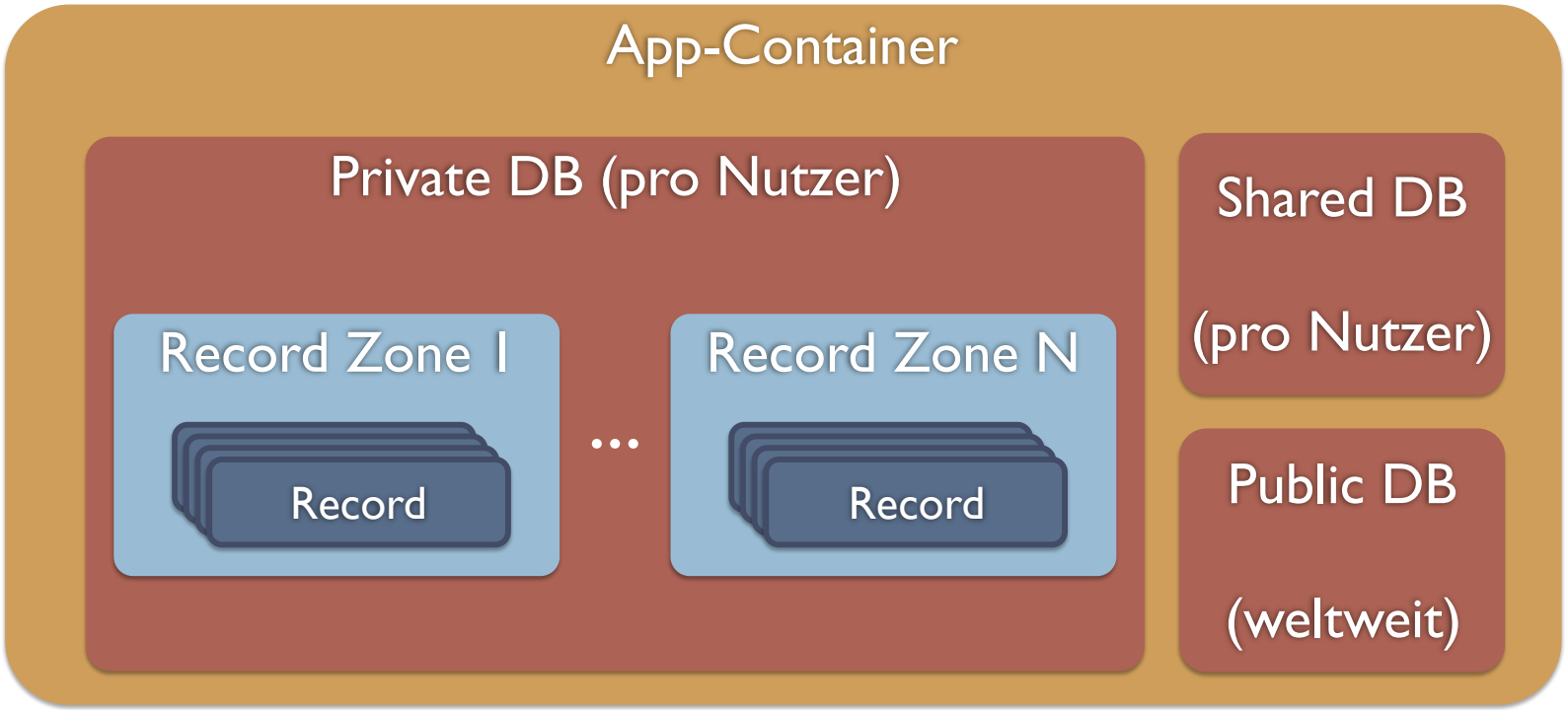
	Dateien	DB	Rx
pro User	100 MB	1 MB	20 MB
Gesamt	1 PB	10 TB	200 TB

Beispiel: Rezeptbuch

Feature #1: Online Sync

- Modellierung der Daten
- Up/Download zu iCloud
- Suche
- Live-Aktualisierung

Die Datenbank(en)

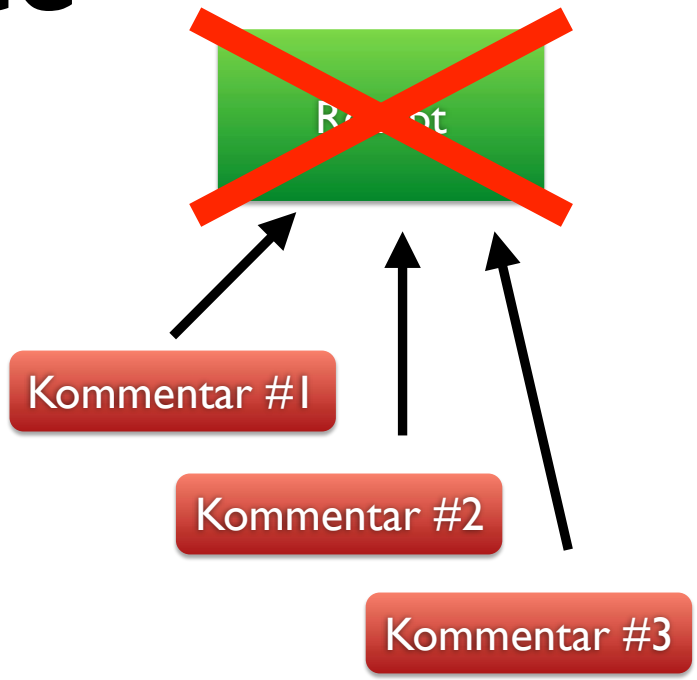


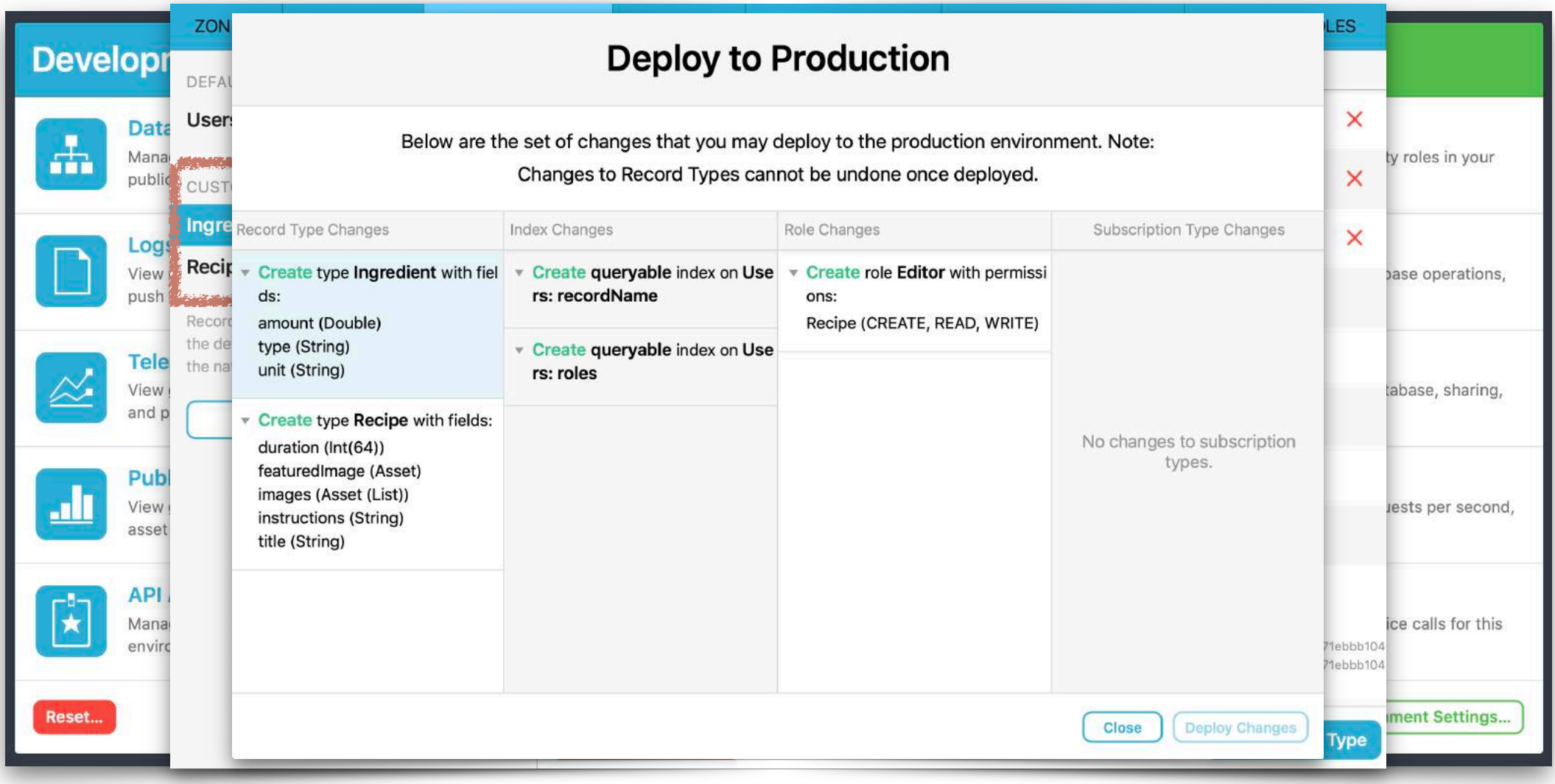
Records

- **Key-/Value-Paare**
- **Identifier**: Record-Name + Zone
- **Record Type:** Schema
    - String, Number, Date, Data
    - CKLocation
    - CKAsset
    - Arrays

CKReference

- Relationen zwischen Objekten
- Referenz von Child zu Parent
- Optional: Kaskadierendes Löschen
- Limit: 750 / Parent




Deploy to Production
Below are the set of changes that you may deploy to the production environment. Note:
Changes to Record Types cannot be undone once deployed.
Record Type Changes
Create type Ingredient with fields:
amount (Double)
type (String)
unit (String)
Create type Recipe with fields:
duration (Int(64))
featuredImage (Asset)
images (Asset (List))
instructions (String)
title (String)
Index Changes
Create queryable index on Users: recordName
Create queryable index on Users: roles
Role Changes
Create role Editor with permissions:
Recipe (CREATE, READ, WRITE)
Subscription Type Changes
No changes to subscription types.
Close
Deploy Changes
Reset...

Up- / Download

- ~~Blockbasierte Convenience-API~~
- NSOperation-basierte API:
  z.B.: `CKFetchRecordsOperation, CKModifyRecordsOperation`
- Quality-of-Service-Optionen
- Long-Lived-Operations

CKFetchRecordsOperation

```
var recordID = CKRecord.ID(recordName: "Kirschkuchen", zoneID: ...)
var operation = CKFetchRecordsOperation(recordIDs: [recordID])


operation.fetchRecordsCompletionBlock = {(records, error) in
    ...
}


CKContainer.default().privateCloudDatabase.add(operation)
```

Fehlerbehandlung

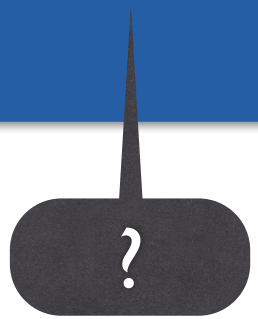
- Ist kritisch…

- **…und** kompliziert: 35 Fehlercodes
  - Temporäre Fehler: networkUnavailable, ...
  - Nutzerfehler: quotaExceeded, notAuthenticated, ...
  - Teilweise-Fehler: Batch-Operationen
  - ...

CKModifyRecordsOperation

```
var op = CKModifyRecordsOperation(recordsToSave: […],
                                  recordIDsToDelete: […])

op.modifyRecordsCompletionBlock = {(savedRecords, deletedIDs, error) in
      …
}
```



Konfliktbehandlung

- Client kennt nicht immer neusten Stand
- Lost Updates…

- **Change Token**: Versionsnummer
- **SavePolicy**: ifUnchanged / changedKeys / allKeys
- **Error**: Konfligierende Version

Suche

- Record IDs meist unbekannt
- **CKQuery:** Suche nach Records mittels NSPredicate
- **CKQueryOperation:** Schrittweises Laden über Cursor
- Erfordert Index

Automatische Aktualisierung

- Suchergebnisse und Records veralten
- Alle Records neu abfragen: Teuer

- `CKSubscription:`
  Persistenter Observer auf Datenbank
- Scope: Datenbank, Zones, Queries
- Push-Notifications bei Änderung

Push Notifications?

- Normale Push Notifications
    - Volle Remote Notifications (Banner, Badge, Sound, ...)
    - Silent Pushes
    - Unkompliziertes Setup - ohne APN
- Payload klein…
- …aber: anschließender Fetch möglich

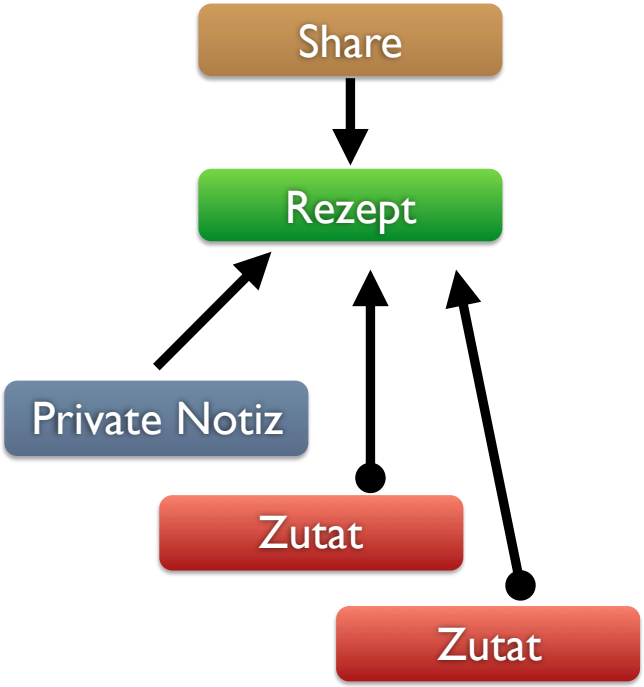
Feature #2: Sharing

- **Shared Database:** Freigebene Records anderer Nutzer
- **Share:** Sicht auf private Records des Besitzers

- Besitzer sendet Einladung:
  User-ID, E-Mail, Telefonnummer
- Empfänger: Verifikation + Bestätigung
- Sharing-UI für mac + iOS

Freigabe von Records

- CKShare: Metadaten + Freigegebener Record
- **Parent Referenz:**
  Freigabe abhängiger Records
- **Andere Referenzen:**
  Records bleiben privat
- Nachträgliches Entziehen Möglich
- Zugriffsrechte und Sichtbarkiet definierbar



Feature #3: Öffentliche Rezepte

- **Public Database**
- Auch ohne iCloud erreichbar
- Quota des Entwicklers (Kosten!)
- Zugriffsrechte & Rollen
- **Besonderheit:** Nutzerdatenbank

Nutzerdatenbank?

- **Communities:** Freunde finden
- User-Datenbank pro App
- Apple: Schutz der Nutzerdaten
- User-Datenbank: Nicht durchsuchbar

Nutzer finden

- Nutzerdatenbank: Nicht direkt durchsuchbar
- Entwickler: kein Zugriff auf Adressen
- CloudKit-Daemon übernimmt Suche:
  - E-Mail-Adresse / Telefonnummer
  - "Ganzes Adressbuch"
- Transfer von Hashes

Web-API

- Vollständige REST-API
- Server-to-Server-API
- CloudKit-JS: Convenience für Web-Apps
- Authentifizierung über API-Tokens
    - User-Accounts: Apple-Website
    - Developer-Account: Zertifikate

Nachteile von CloudKit

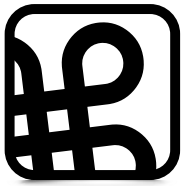
- Keine Server-seitige Logik (Mischlösungen?)
- Kein Sync-Framework
- Kein Modellierungsframework
- Integrationtests schwierig
- **Support:** Wer ist Apples Kunde?
- Undokumentierte Limits

Fazit

- CloudKit: BaaS von und für Apple
- Günstiger Walled Garden mit Vorplatz
- Gut für Apps mit Nutzerdaten + öffentliche Daten
- Skaliert nicht in alle Bereiche (Serverlogik, …)
- WWDC-Talks sehr empfehlenswert!

Fragen?

Vielen Dank



Macoun