

---

# INGREDISCAN

Deine smarte Lupe für Lebensmittel.

---

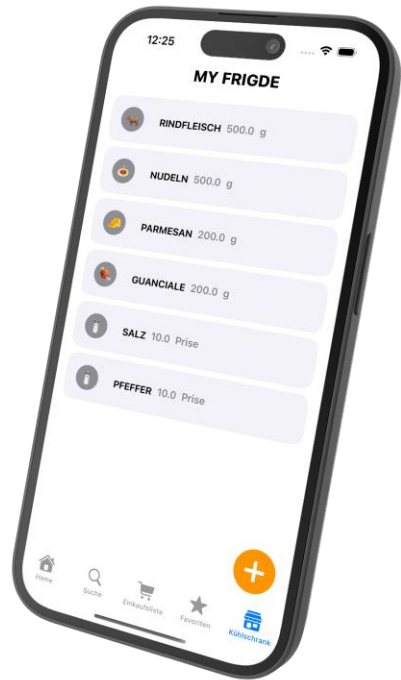
Jan Keller, Fynn Schotten (Gruppe 4)



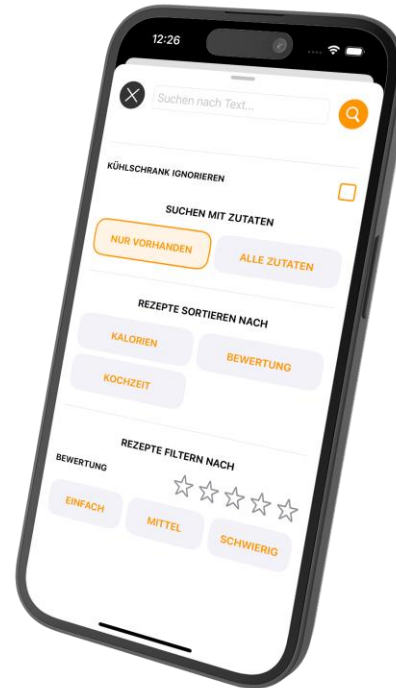
IDEE

Was soll ich heute kochen?

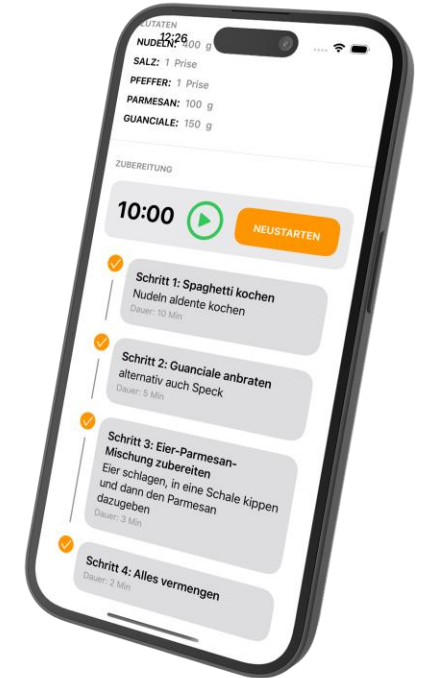
# LÖSUNG: INGREDISCAN



Kühlschrank



Suchen



Kochen

---

# DEMO



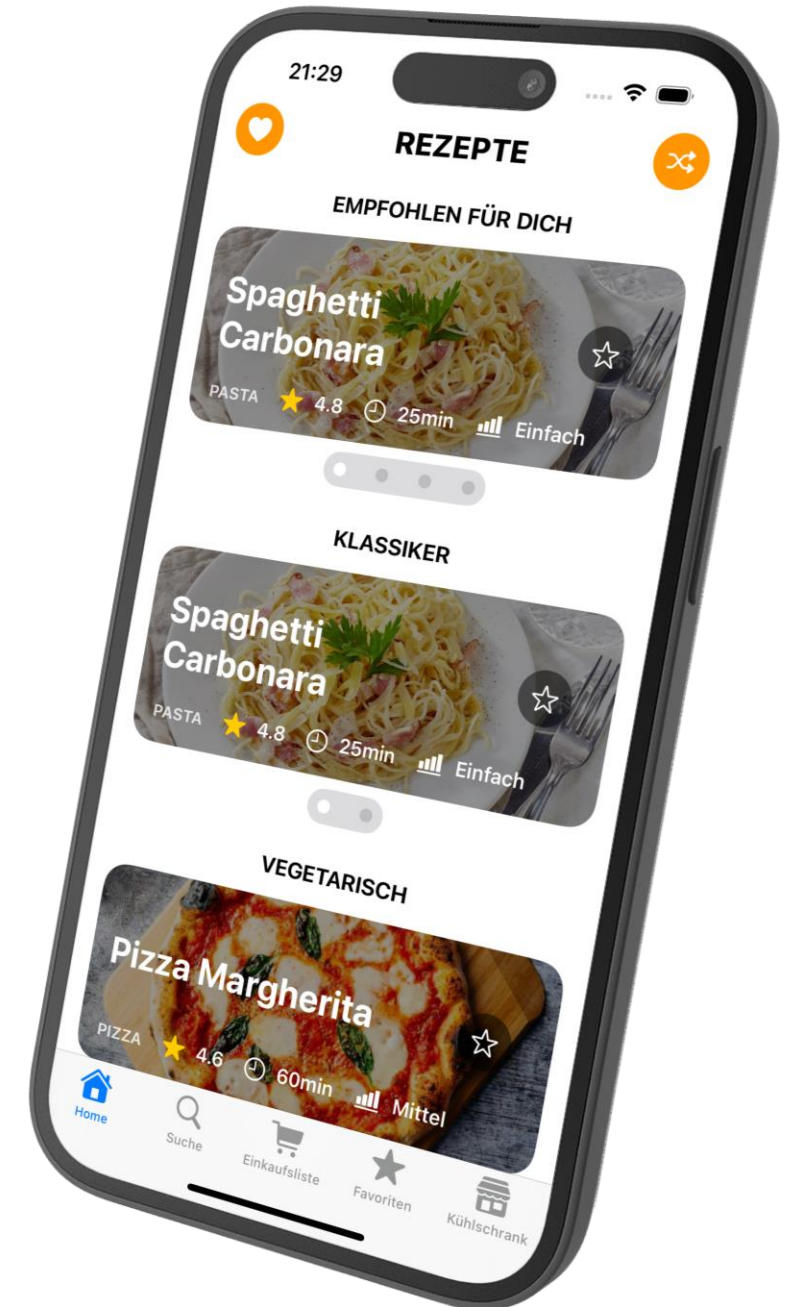
**IngrediScan**

---

# ÜBERSICHT

- Inspiration für den Nutzer
- Durch die Rezepte stöbern
- Rezepte in ihren Kategorien

```
func loadRecipes() async {  
    self.recipes = await DatabaseService.shared.fetchRecipes()  
}  
  
func loadTags() async {  
    self.tags = await DatabaseService.shared.fetchTags()  
}  
  
func loadCategories() async {  
    self.categories = await DatabaseService.shared.fetchCategories()  
}  
  
func loadIngredients() async {  
    self.ingredients = await DatabaseService.shared.fetchIngredients()  
}  
  
func loadFavorites() {  
    favoriteIDs = UserDefaults.standard.array(forKey: favoritesKey) as? [Int] ?? []  
}
```



---

# VIRTUELLER KÜHLSCHRANK

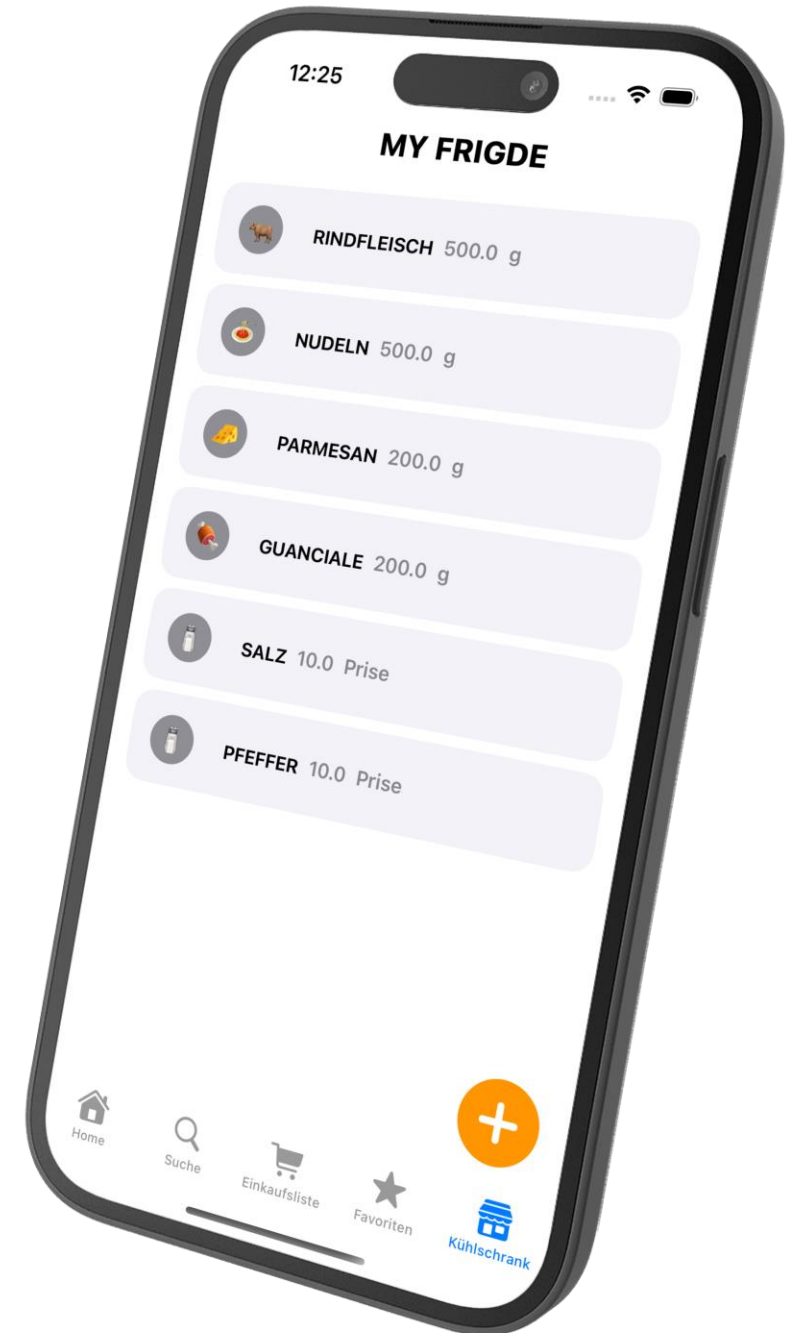
- Kühlschrank in digitaler Form
- Einfache Pflege
- Datengrundlage

```
class FridgeViewModel: ObservableObject {
    @Published var items: [FridgeItem] = []

    private let itemsKey = "fridgeIngredientsKeys"

    init() {
        loadItems()
    }

    func addItem(_ item: FridgeItem) {
        if !items.contains(where: { $0.ingredientId == item.ingredientId }) {
            items.append(item)
        } else {
            modifyItem(item)
        }
        saveItems()
    }
}
```





# SUCHE

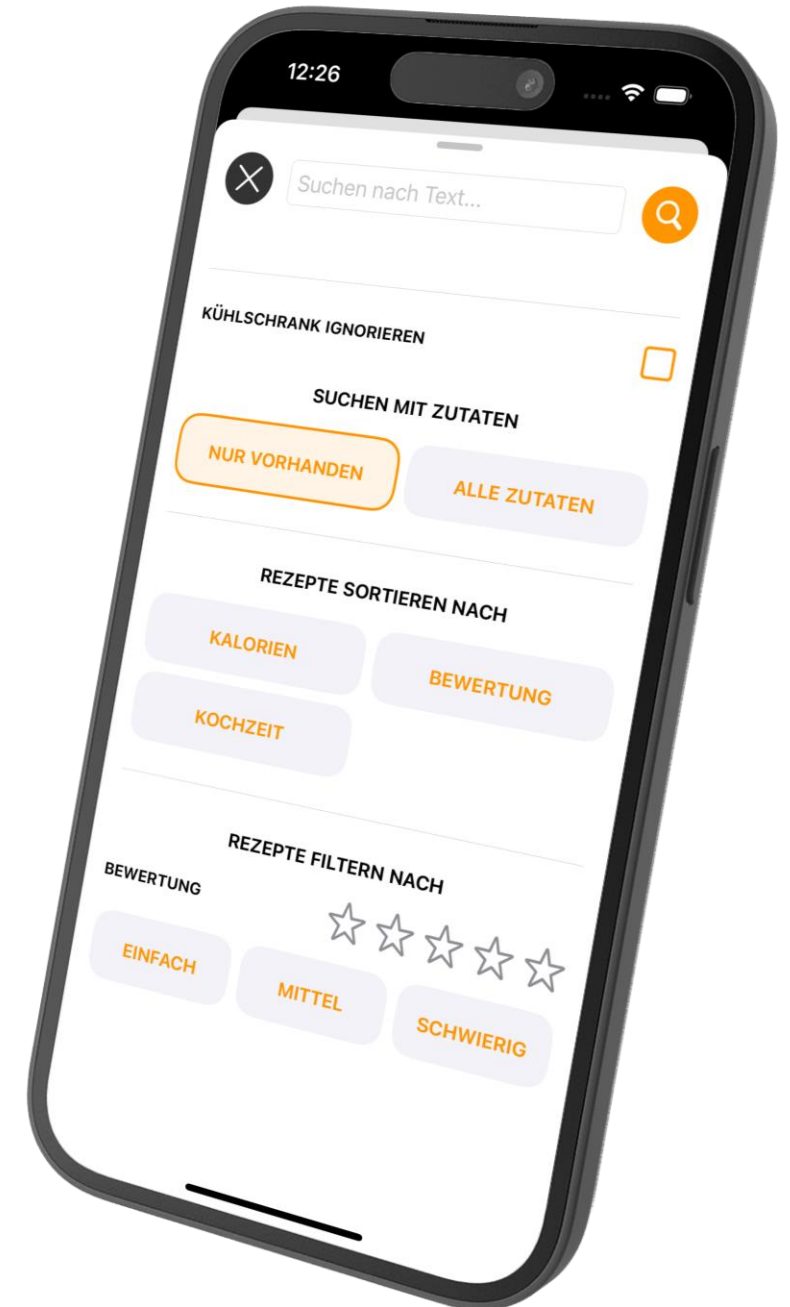
- Rezepte sortieren und filtern
- Zugeschnittenes Ergebnis auf den Nutzer
- Viele Anwendungen

```
//sort the result
let sortByRating = selectedSortCategories.contains("Bewertung") ? 1 : 0
let sortByCalories = selectedSortCategories.contains("Kalorien") ? 1 : 0
let sortByDuration = selectedSortCategories.contains("Kochzeit") ? 1 : 0

searchResult = searchResult.sorted {
    let rating1 = ($0.rating) * Float(sortByRating) * 200
    let calories1 = Float($0.calories) * Float(sortByCalories)
    let duration1 = Float($0.duration) * Float(sortByDuration) * 10
    let score1 = rating1 - calories1 - duration1

    let rating2 = ($1.rating) * Float(sortByRating) * 200
    let calories2 = Float($1.calories) * Float(sortByCalories)
    let duration2 = Float($1.duration) * Float(sortByDuration) * 10
    let score2 = rating2 - calories2 - duration2

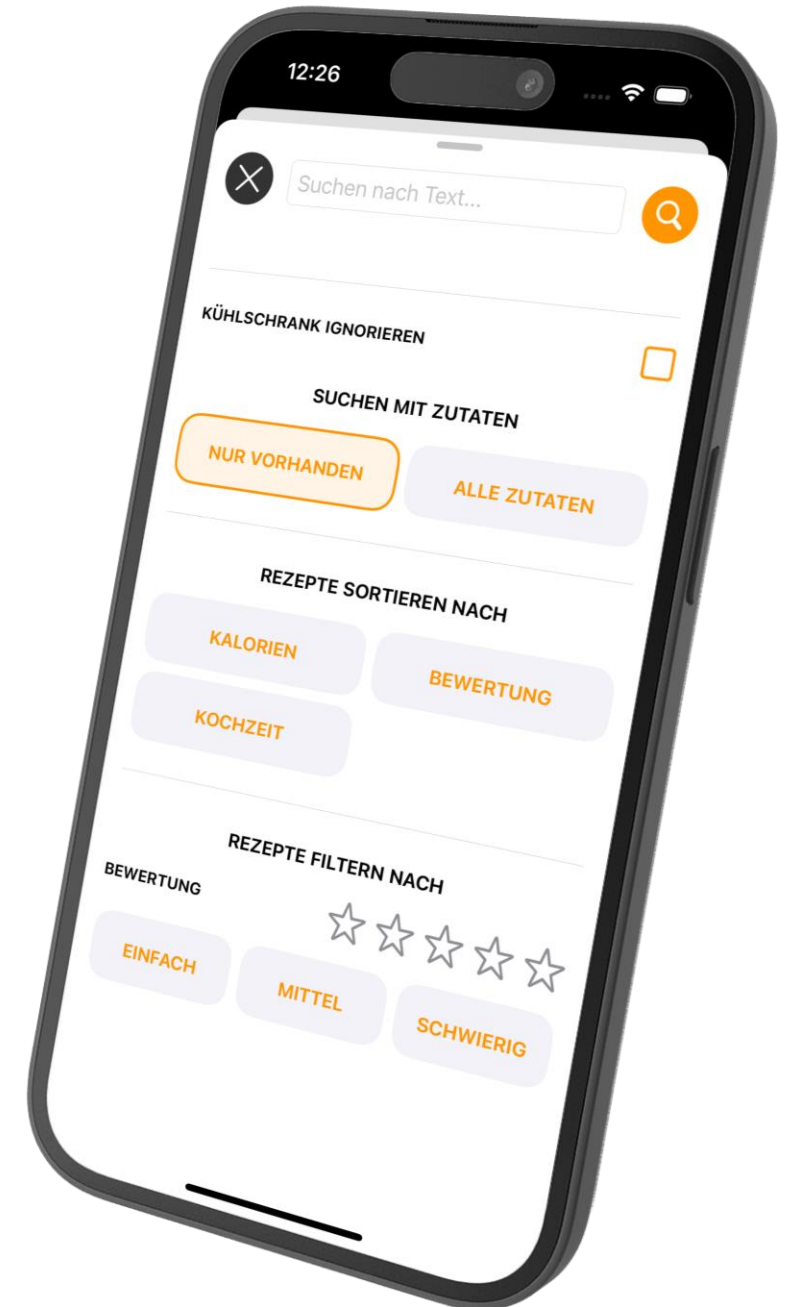
    return score1 > score2
}
```



# KOCHEN

- Timer in der App
- Übersicht beim Kochen
- Nichts vergessen

```
class TimerViewModel: ObservableObject {  
    init(initialDuration: TimeInterval = 10) {  
        totalDuration = initialDuration  
        remainingTime = initialDuration  
  
        self.timer = AdjustableTimer(duration: initialDuration, onTick: { [weak self] timeLeft in  
            DispatchQueue.main.async {  
                self?.remainingTime = timeLeft  
            }  
        }, onComplete: { [weak self] in  
            DispatchQueue.main.async {  
                self?.isRunning = false  
                VibrationService.shared.vibrateForSeconds(5)  
            }  
        })  
    }  
  
    func start() {  
        timer?.start()  
        isRunning = true  
    }  
}
```



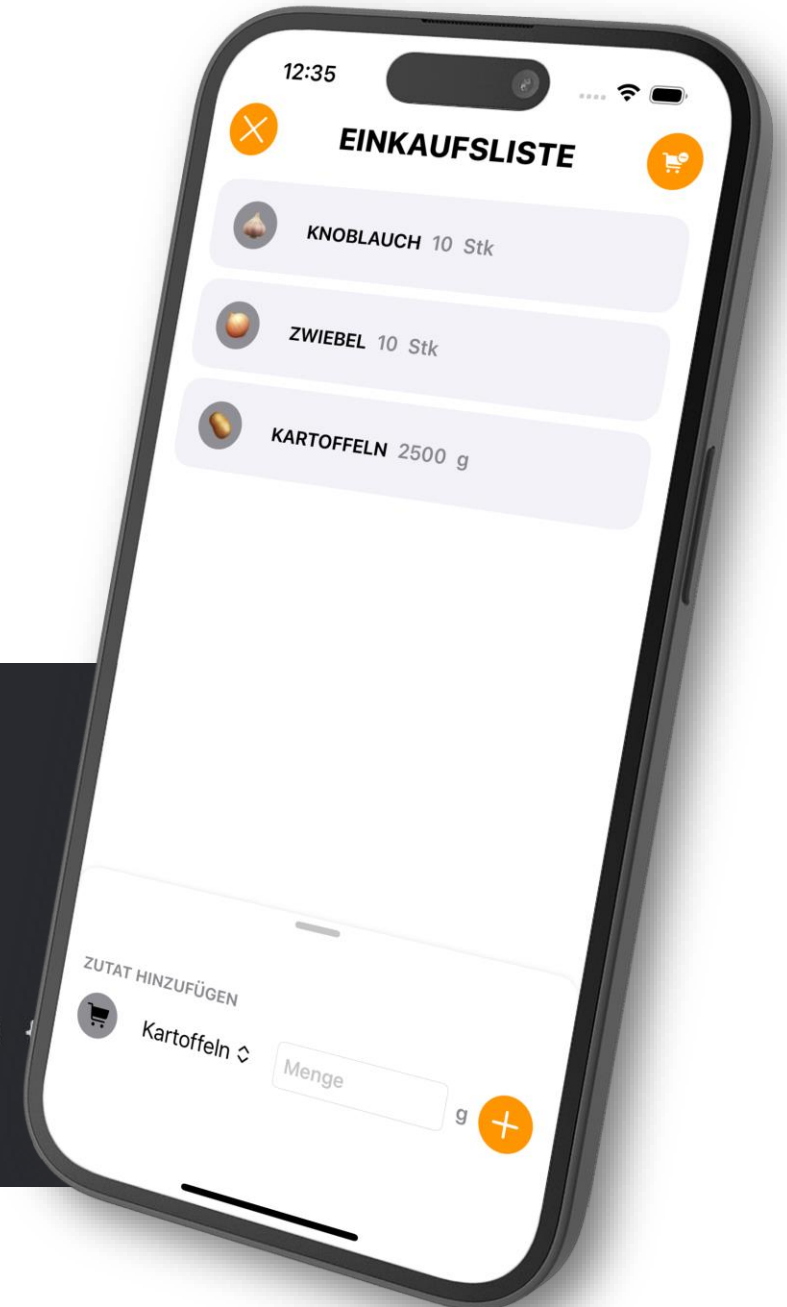


---

# EINKAUFSLISTE

- Intuitiv
- Übersichtlich
- User Defaults für Persistenz

```
private func saveItems() {  
    if let data = try? JSONEncoder().encode(items) {  
        UserDefaults.standard.set(data, forKey: itemsKey)  
    }  
}  
  
private func loadItems() {  
    if let data = UserDefaults.standard.data(forKey: itemsKey),  
       let decoded = try? JSONDecoder().decode([ShoppingItem].self, from: data) {  
        items = decoded  
    }  
}
```



---

# AUFTEILUNG

## **Beide**

- Datenbank
- MockUps

## **Jan**

- Rezeptsuche
- Virtueller Kühlschrank
- UserDefaults (Kühlschrank)

## **Fynn**

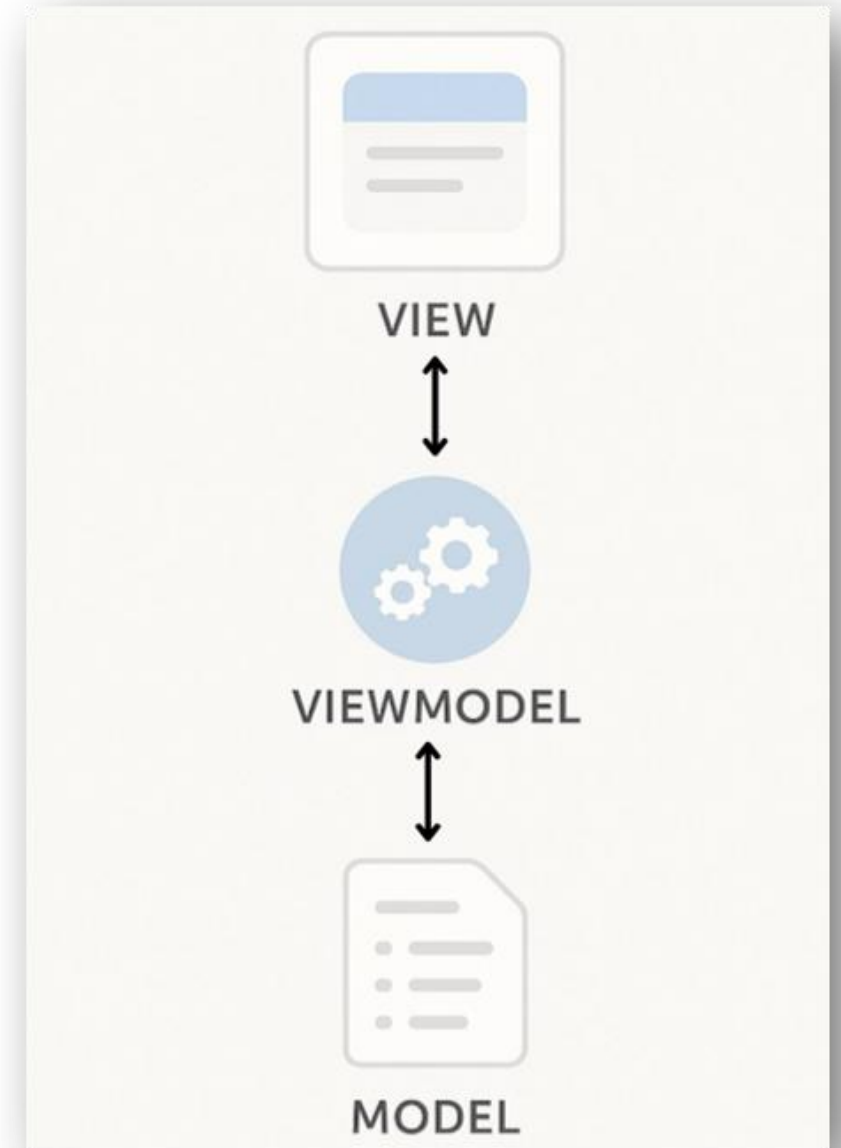
- Rezeptübersicht
  - Kochfunktion
  - UserDefaults (Favoriten, Einkaufsliste)
-

---

# ARCHITEKTUR: MVVM

Trennung von UI, Logik & Daten

- **Model** repräsentiert Daten
- **View** Darstellung der UI
- **ViewModel** Businesslogik & Vermittler



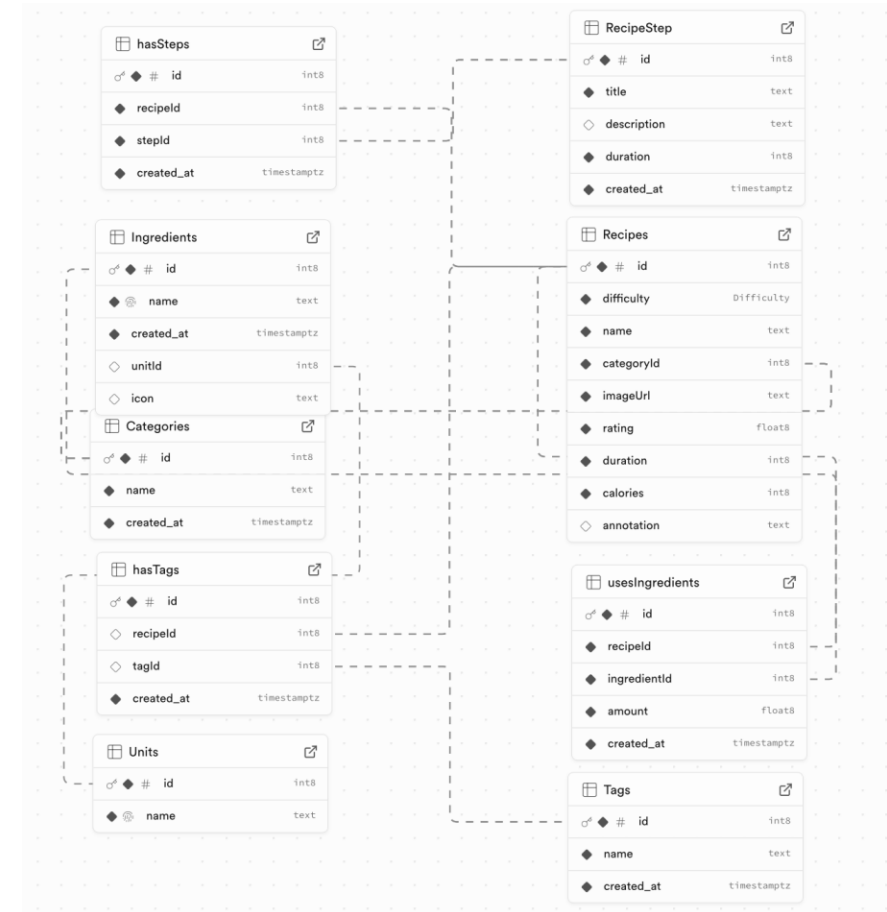
# ARCHITEKTUR: MODEL



Supabase



UserDefaults



---

# ARCHITEKTUR: VIEW-MODEL

Business Logik

Dependency Injection

Bindeglied

Kommunikation mit Services

---

---

# ARCHITEKTUR: VIEW

- Komponenten
- Gekapselt
- Kompakt
- Wiederverwendbar

```
fileprivate struct CardBackground: View {
    @Binding var recipe: Recipe
    @State var isLoading: Bool = false

    var body: some View {
        WebImage(url: URL(string: self.recipe.imageUrl)) { image in
            image
                .resizable()
                .scaledToFit()
                .aspectRatio(contentMode: .fill)
                .frame(height: 150)
                .clipped()
                .overlay {
                    RoundedRectangle(cornerRadius: 16)
                        .foregroundColor(.black.opacity(0.4))
                }
        } placeholder: {
            CardImagePlaceholder()
        }
        .indicator(.activity)
        .transition(.fade(duration: 0.5))
    }
}
```



```
struct RecipeCard: View {
    @State var recipe: Recipe
    @State var detailedView: Bool = false

    var body: some View {
        ZStack(alignment: .trailing) {

            Button {
                detailedView.toggle()
            } label: {
                ZStack(alignment: .leading) {
                    CardBackground(recipe: $recipe)
                    CardText(recipe: $recipe)
                        .padding(.leading)
                }
            }
            .cornerRadius(20)
            .fullScreenCover(isPresented: $detailedView) {
                RecipeDetailView(recipe: self.$recipe)
            }
        }
        .padding()
    }
}
```



---

# LEARNING

- Simulator ist nur ein Simulator
- Projektplanung war gut
- SwiftUI sehr mächtig „Out of the Box“
- Viel Code im Internet ist veraltet
- Schnelle visuelle Entwürfe -> Details zeitaufwendig
- Viele Features haben sich im Verlauf geändert oder erst entwickelt
- Supabase ist einfach einzubinden
- Merge Konflikte in anderer IDE resolve



---

# IN THE FUTURE



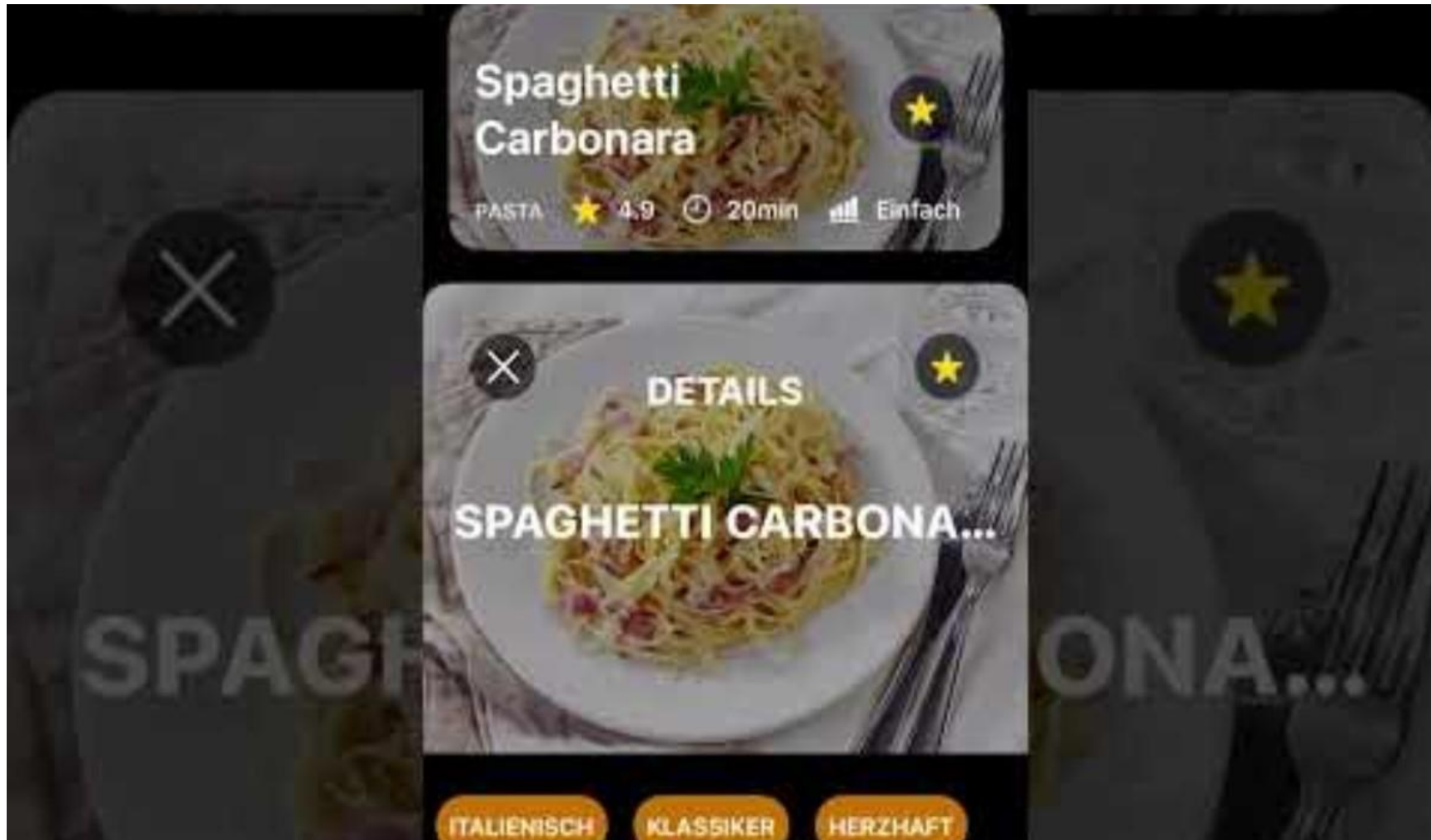
- Accounts
- Erstellung eigener Rezepte
- Bewertung von Rezepten
- Einkaufszettel-Scan
- Essensplan für die Woche
- Preis für Zutaten
- Intelligente Algorithmen für Rezeptübersicht
- Veröffentlichung





---

VIELEN DANK



---

FALLBACK