BudgIT: Android Budget App

Abschlusspräsentation Mobile Anwendungen

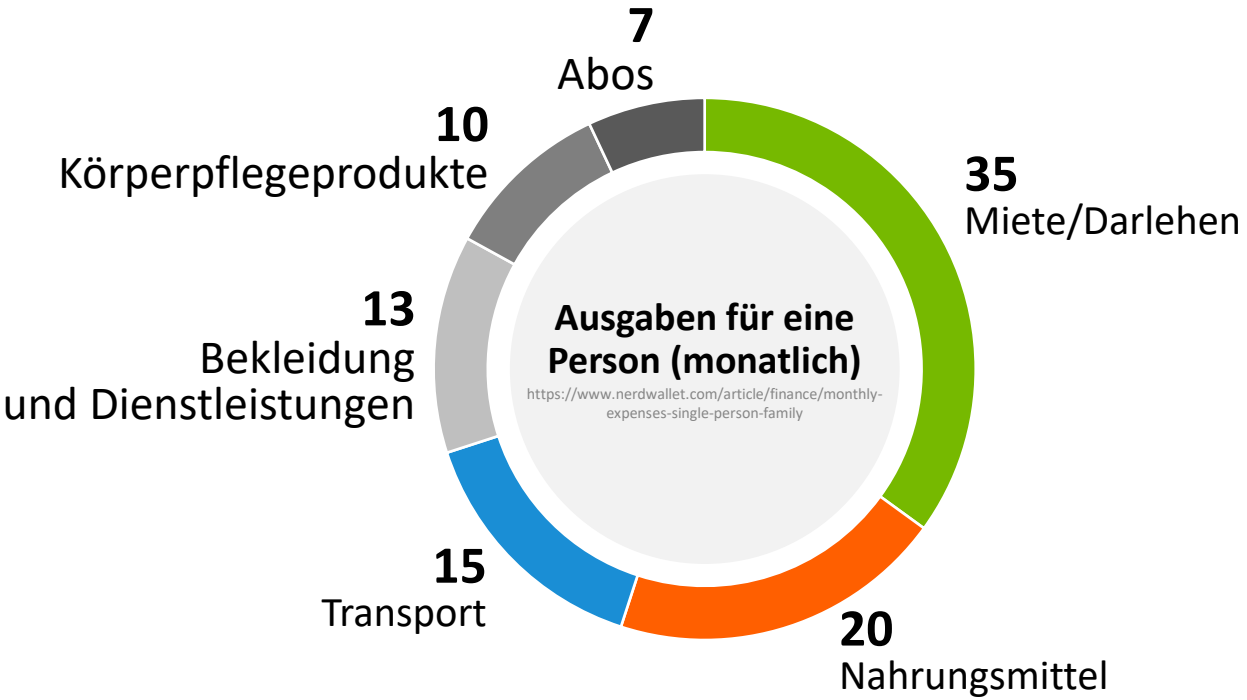
Vladimir Osetrov und Jan Furio / 28.01.2024

Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences

Warum *noch* eine Budget App?

Werte in Prozent



Warum *noch* eine Budget App?

https://www.businessofapps.com/data/google-play-statistics/

2.65Milliarden
Apps auf Google Play Store

ca. 90%
Nicht FOSS

Low weight
zu wenige

ca. 76%
Google AdMob u.a.

80/20 Rule
80% der Apps gehören 20% der Developer

Offline?
Fasst alle Apps verlangen Zugriff auf mobile Daten

Ziele der BudgIT App

Minimale UI

BudgIT verwendet das Material UI 3.0 von Google, um ein einheitliches Design zu gewährleisten.

Offline-Funktionalität

Durch die Verwendung der Room Database bleiben alle Nutzerdaten sicher auf dem Gerät gespeichert.

Einfache Navigation

Die App verfügt über eine Navigationsleiste am unteren Bildschirmrand, die eine intuitive Bedienung ermöglicht.

FOSS

Der vollständige Code von BudgIT wird auf GitLab veröffentlicht, um das Engagement für FOSS zu demonstrieren und der Community etwas zurückzugeben.

Budgeting

BudgIT zielt darauf ab, den Nutzern zu helfen, ihre Finanzen effektiver zu verwalten.

Funktionale Anforderungen der BudgIT App

Historie

Integrierte Transaktionhistorie, welche manuell angegeben wird: „Label“ haben und ein „Amount“.

Fragmente

Die App sollte ein Home Fragment, ein Add Transaction Fragment, ein Budget Fragment und ein Transaction Fragment besitzen.

Budget

Budget manuell in der App einzugeben. Jede neue Eingabe: Amount und Frequenz („daily“, „weekly“, „fortnightly“, „monthly“, „yearly“) angegeben werden können.

Infrastruktur

Die App speichert die Transaction und Budget innerhalb der RoomDB

Home

Die Homescreen sollte einen klaren überblick des finanziellen Stands des Users zeigen.

Mit 3 der letzten Transaktionen.

Fragmente der BudgIT App

Home

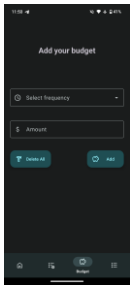
Übersicht über Guthaben, Ausgaben und Budget.

Add Transaction

Eingabeformular für neue Transaktionen.

Budget

Einstellungsoptionen für das Budget.



Transaction

Liste der getätigten Transaktionen mit Löschfunktion.

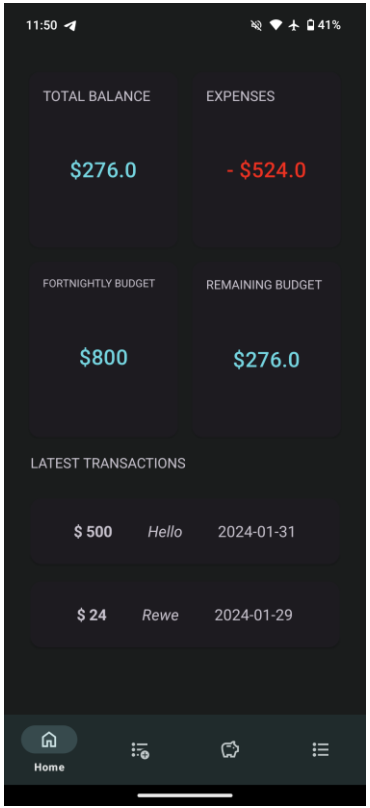
Home

Im Überblick

Zeigt die Gesamtbilanz, die Ausgaben, das Budget und das verbleibende Budget des Benutzers an

Letzte Transaktionen

Zeigt die letzten Transaktionen (Datum desc.)



Add Transaction

Label

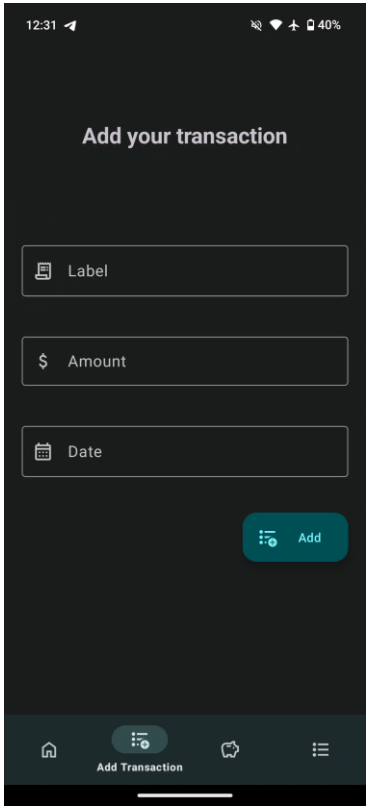
User gibt als String den Transaktionsnamen ein

Amount

User gibt als Double den Betrag ein

Date

Zeigt eine von Material 3.0 Datum Pop-Up, dieser wird als DateTime Typ eingegeben



Budget

Frequenz auswählen

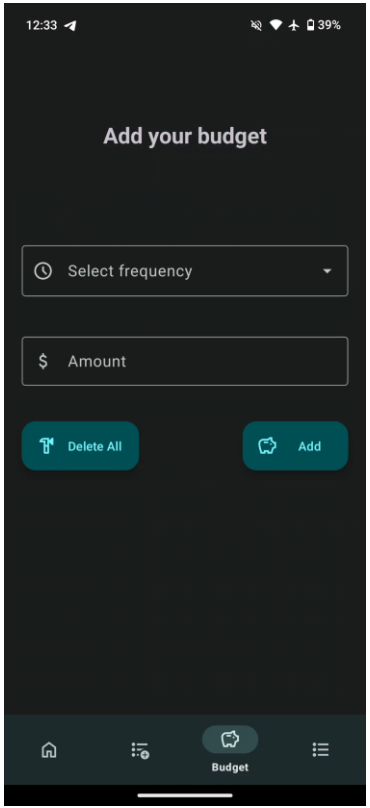
User wählt aus dem dropdown-Kasten zwischen: daily, weekly, fortnightly, monthly oder yearly aus

Amount

User gibt als Double den Betrag seines neuen Budgets ein

Delete All

User kann all seine Eingegebenen und automatisch generierten Budgets löschen



Transactions

Transaction History

Alle von den User hinzugefügten Transaktionen werden hier angezeigt

Swipe to Delete

User kann Transaktionen einzeln durch rechts oder links swipe Geste löschen

Delete All

User kann alle von ihm eingegebene Transaktionen löschen

Datenbank Strutur

budget Tabelle

Besitzt eine PrimaryKey, speichert die frequency als String und den amountBudget als Int

```
@Entity(tableName = "budget")
data class UserBudget(
    @PrimaryKey(autoGenerate = true) val id:Int = 0,
    val frequency: String,
    val amountBudget: Int
)
```

Transactions Tabelle

Besitzt eine PrimaryKey, speichert den amount, label und date als String

```
@Entity(tableName = "transactions")
data class UserTransaction(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val amount: String,
    val label: String,
    val date: String
)
```

Vielen Dank.

www.htw-berlin.de

Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

www.htw-berlin.de