

Budget App - Gesällprov

Anna Strömberg, anst5816

Institutionen för data-
och systemvetenskap

Vetenskaplig metodik och kommunikation inom data- och
systemvetenskap

Gesällprov rapport

Programmering för Mobiler

Vårterminen 2025



Stockholms
universitet

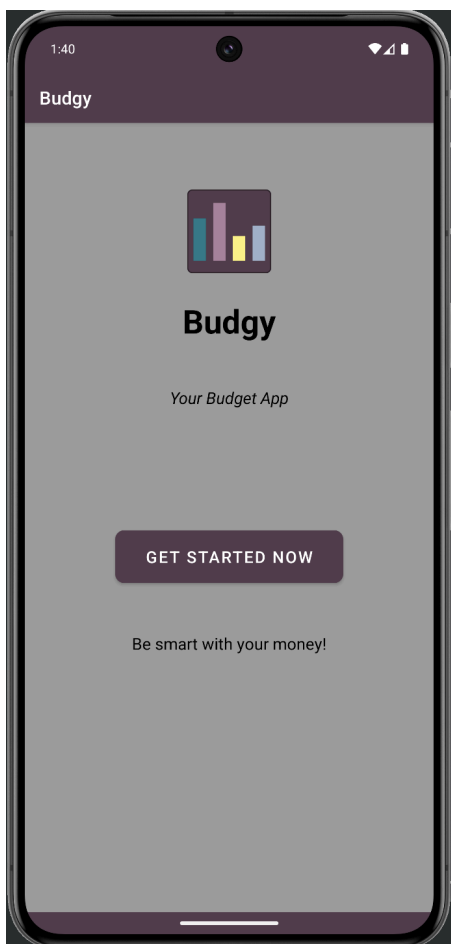
Innehållsförteckning

Sammanfattning	2
Framtagande	3
Idé och mål	3
Framtagande av wireframes i Figma	3
Framtagande av fullständig design i Figma	9
Kodning av grundstruktur med Java	15
Kodning av stil och interaktivitet med XML	16
Förändring från skiss till färdig produkt	16
Grundläggande struktur (Java)	19
Stil och layout (XML)	20
Viktiga kodavsnitt	21
Helheten vid vanligt use case	31
Funktion	33
Användarens första interaktion	33
Inkomstmatning	33
Hantering av utgifter	34
Sammanfattning och analys	38
Navigering och interaktion	40

Sammanfattning

Budgetappen Budgy är en intuitiv och användarvänlig applikation som hjälper användare att få en tydlig överblick över sin ekonomi. Syftet med appen är att förenkla budgethantering genom att visualisera inkomster och utgifter i en interaktiv PieChart. Användaren kan mata in sin inkomst och kategorisera sina utgifter i fasta, rörliga och lån/krediter. Baserat på denna information beräknar appen balansen och presenterar en översikt på en sammanfattningssida. Med en navigeringsmeny kan användaren enkelt gå tillbaka och justera sin ekonomi och genom en tydlig och stilren design gör Budgy det enkelt att hålla koll på sin ekonomi på ett visuellt och pedagogiskt sätt.

Bild från startsidan av appen:



Framtagande

Utvecklingen av min app Budgy gick igenom flera steg, bland annat använde jag både designverktyget Figma och Android Studio som utvecklingsmiljö. Detta för att skapa en funktionell och estetiskt tilltalande app.

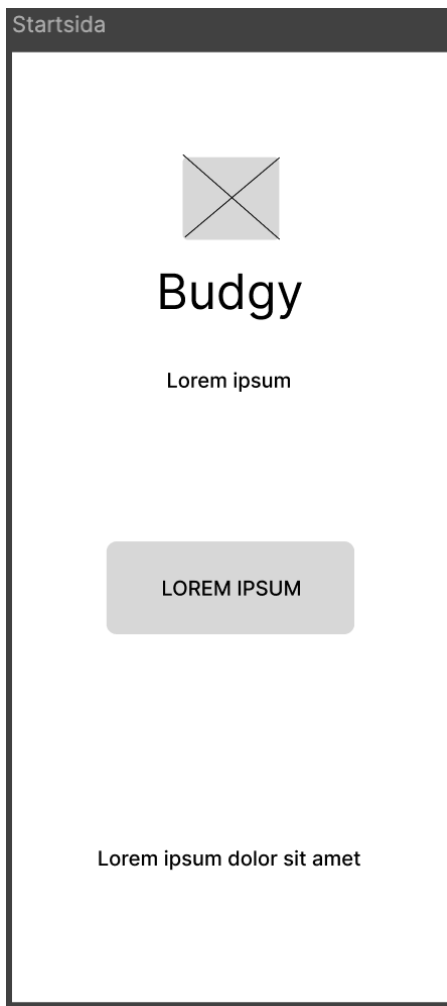
Idé och mål

Jag visste direkt när jag såg uppgiften att jag ville skapa en budgetapp. Jag blev inspirerad att skapa denna app efter att ha sett flera avsnitt av TV-programmet Lyxfällan och insett hur viktigt det är att ha en tydlig överblick över sin ekonomi. Eftersom jag själv också har ett stort intresse för ekonomi och budgetering ville jag utveckla ett verktyg som gör det enkelt att hantera sin ekonomi på ett visuellt och användarvänligt sätt. Målet var att skapa en budgetapp där användaren kan mata in sin inkomst och sina utgifter, och sedan få en sammanfattning i form av en interaktiv PieChart. Jag ville också att appen skulle ha en logisk navigering, där användaren enkelt kan gå tillbaka och justera sina uppgifter.

Framtagande av wireframes i Figma

För att skapa en tydlig struktur och planera användarupplevelsen började jag med att skissa wireframes i Figma. Jag fokuserade på att designen skulle vara enkel, överskådlig och lätt att navigera. Wireframsen består av enkla svartvita skisser av appen, genom att använda denna designstrategi kunde jag experimentera med olika layouter och testa placeringar av de olika elementen utan att tänka på detaljer som färger eller bilder i det initiala skedet.

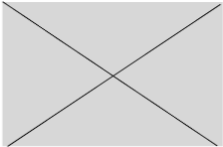
Startsida Wireframe



Inkomst Wireframe

Inkomst

WELCOME
Let's set up your budget!



Lorem ipsum
Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do
eiusmod tempor incididunt

KR

NEXT

Fasta utgifter Wireframe

Fasta utgifter

ADD FIXED EXPENSES

Expense Name

Amount

ADD

Added Expenses:

Lorem ipsum	1000 KR	✕
Lorem ipsum	1000 KR	✕
Lorem ipsum	1000 KR	✕
Lorem ipsum	1000 KR	✕

Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do
eiusmod tempor incididunt

NEXT

Lån/Krediter Wireframe

Lån/Krediter

ADD LOANS AND CREDITS

Expense Name

Amount

ADD

Added Expenses:

Lorem ipsum	100 000 KR	✕
Lorem ipsum	10 000 KR	✕
Lorem ipsum	1 000 000 KR	✕
Lorem ipsum	1000 KR	✕

Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do
eiusmod tempor incididunt

NEXT

Rörliga utgifter Wireframe

Rörliga utgifter

ADD VARIABLE EXPENSES

Expense Name

Amount

ADD

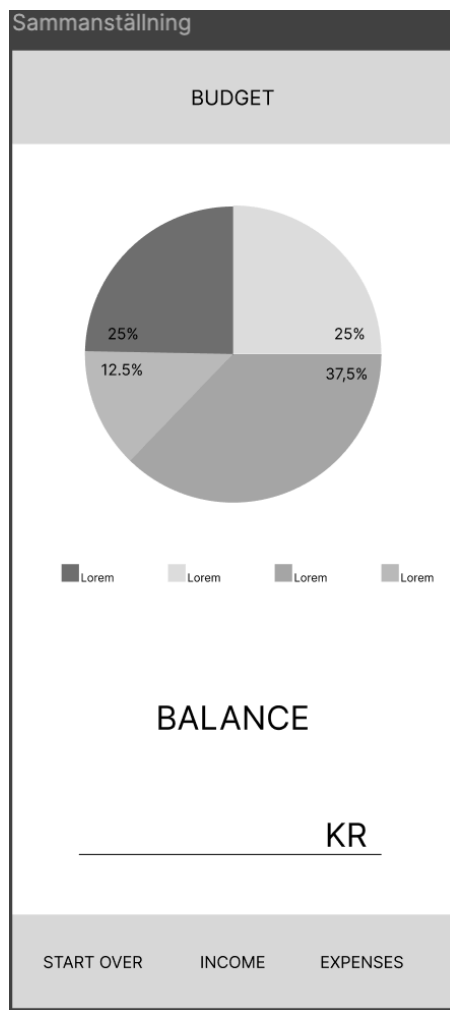
Added Expenses:

Lorem ipsum	1000 KR	✕
Lorem ipsum	1000 KR	✕
Lorem ipsum	1000 KR	✕
Lorem ipsum	1000 KR	✕

Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do
eiusmod tempor incididunt

MAKE BUDGET

Sammanställning Wireframe



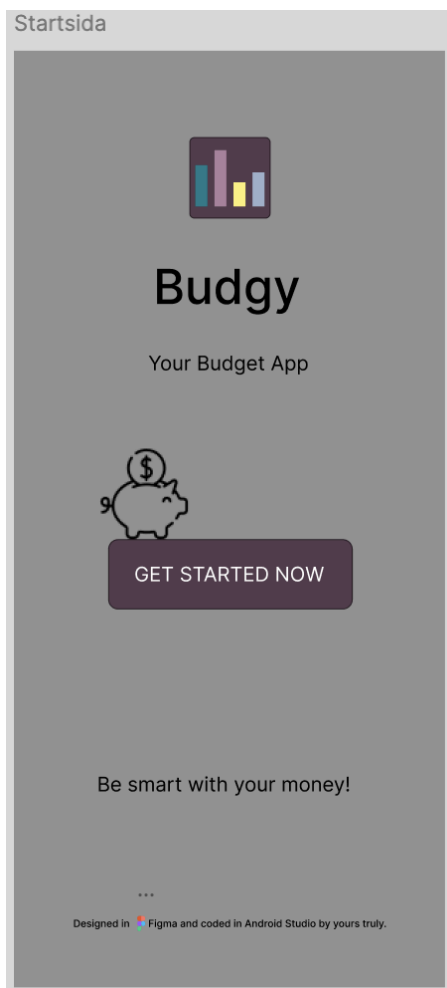
Framtagande av fullständig design i Figma

När wireframes var klara gick jag vidare med att skapa den faktiska designen. Här valde jag färger, typsnitt och layout som skulle ge en modern och professionell känsla.

Färgschema:

- Primärfärg: Mörklila (för att ge en elegant och seriös känsla)
- Sekundärfärger: Gult, ljusblått, rosa och petrol (för att framhäva viktiga knappar/element)
- Bakgrund: Ljusgrått (för att skapa kontrast och en stilren layout)


Startsida Fullständig design



Inkomst Fullständig design

Inkomst

WELCOME
Let's set up your budget!



INCOME

Income includes: Salary,
Allowance, Dividend, Profit and
Interest

37 500 KR

NEXT

Fasta utgifter Fullständig design

Fasta utgifter





ADD FIXED EXPENSES

Expense Name

Amount

ADD

Added Expenses:

Rent	7000 KR	
Insurance	1200 KR	
Gym membership	550 KR	
Netflix subscription	199 KR	

TOTAL: 8 949 KR

Fixed expenses are costs in your budget that do not vary from month to month, such as: Rent, Insurance, Membership/Subscription costs, Phone bill etc.

NEXT

Lån/Krediter Fullständig design

Lån/Krediter





ADD LOANS AND CREDITS

Expense Name

Amount

ADD

Added Expenses (monthly payment):

Car loan	5500 KR	
Student loan	750 KR	
House loan	11 700 KR	
Klarna	100 KR	

TOTAL: 18 050 KR

Loan and credits include: Car loans, Personal loans, Student loans, Credit bills and loans etc.

NEXT

Rörliga utgifter Fullständig design

Rörliga utgifter

ADD VARIABLE EXPENSES

Expense Name

Amount

ADD

Added Expenses:

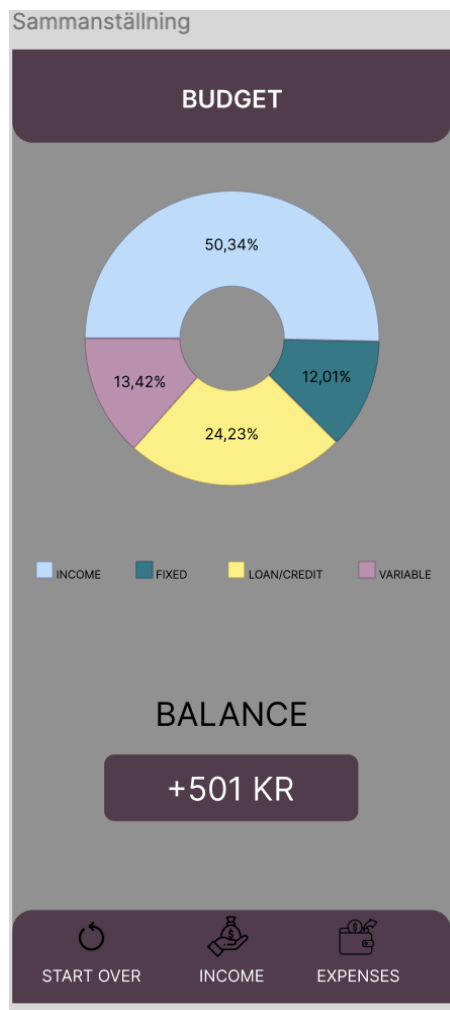
Gas	2000 KR	
Groceries	4000 KR	
Shopping	3000 KR	
Foodora	1000 KR	

TOTAL: 10 000 KR

Variable expenses are costs that
change over time, such as:
Groceries, Gas, Dining out,
Hygiene articles, Pet expenses
etc.

MAKE BUDGET

Sammanställning Fullständig design



Kodning av grundstruktur med Java

Efter att den fullständiga designen var klar och jag kände mig nöjd började jag koda appen. Jag använde mig av Java för att bygga funktionaliteten i appen.

Steg 1: Implementera budgetlogik

- Skapa klassen "Expense.java" som lagrar och beräknar budgetuppgifter.
- Skapa klassen "ExpenseList.java" som visar listan av utgifter användaren matat in.
- Implementera metoder för att spara och hämta inkomster och utgifter.

Steg 2: Skapa aktiviteter för varje skärm

- Start.java - startskärmen
- Income.java – för inmatning av inkomst.
- FixedExpense.java – för fasta utgifter.
- LoanAndCredit.java - för lån och krediter
- VariableExpense.java – för rörliga utgifter.
- Summary.java – för att sammanfatta budgeten med en PieChart.

Steg 3: Implementera navigation mellan aktiviteter

- Använda Intent för att skicka användaren mellan de olika sidorna.
- Implementera `setOnClickListener` på knapparna för att navigera i appen.

Steg 4: Implementera PieChart med EazeGraph

- PieChart genererades baserat på de inmatade värdena.
- Färger och etiketter anpassades för att skapa en tydlig och överskådlig vy.

Kodning av stil och interaktivitet med XML

När grundfunktionaliteten var på plats gick jag vidare till att förbättra design och interaktivitet med hjälp av XML.

Steg 1: Layout för varje skärm

- Skapa en ren och responsiv layout med `ConstraintLayout`.
- Lägga till knappar, `EditText`-fält, `TextViews` och `PieChart`.

Steg 2: Anpassa knappar och navigation

- `Button`-element med `android:drawableTop` för att ha ikon + text.
- En dynamisk knapp (`back_to_chart`), som endast visas om användaren kommer från `Summary`.

Steg 3: Implementera dynamisk synlighet

- I `Income.java` hanterades synligheten av knappen `backToChartButton`, så att den endast visas om användaren navigerat från `Summary.java`.

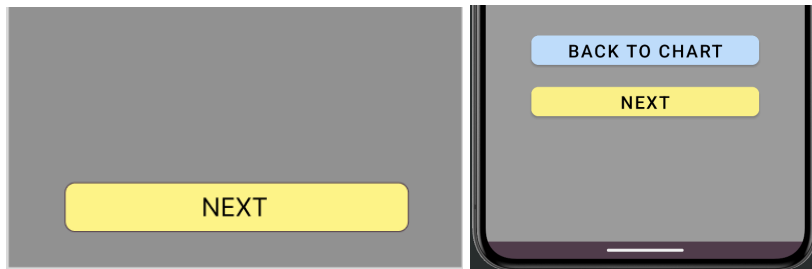
Förändring från skiss till färdig produkt

Under utvecklingen gjordes flera justeringar från den ursprungliga skissen till den färdiga produkten:

1. Lagt till en extra knapp på Income-sidan

- Tidigare behövde användaren gå igenom hela budgetprocessen igen, nu kan den snabbt gå tillbaka till `Summary` efter att ha ändrat sin inkomst.

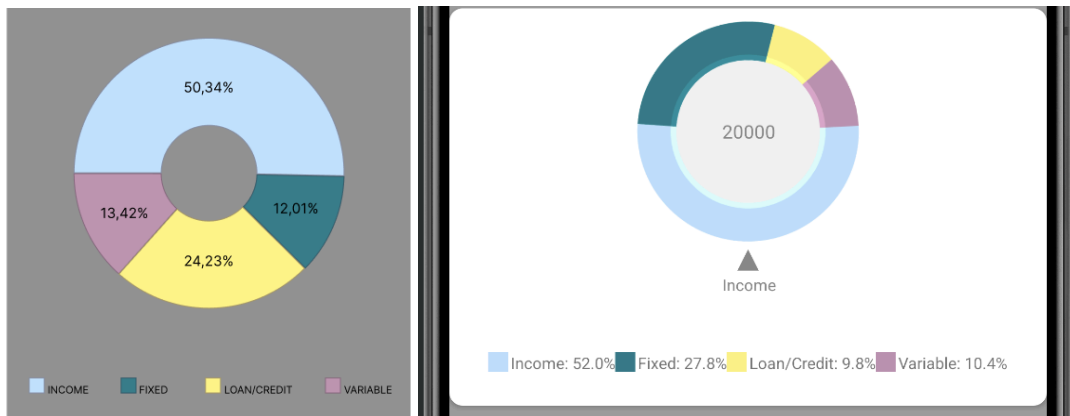
Design (till vänster) - Slutgiltigt (till höger)



2. Visuell förbättring av PieChart

- Ändrade designen lite och lade även det procentuella bredvid kategorierna nedanför piecharten istället för i.

Design (till vänster) - Slutgiltigt (till höger)



3. Dynamisk synlighet

- Knappen "Back to PieChart" syns endast om användaren har tidigare lagt in sin inkomst, vilket förbättrar användarupplevelsen.

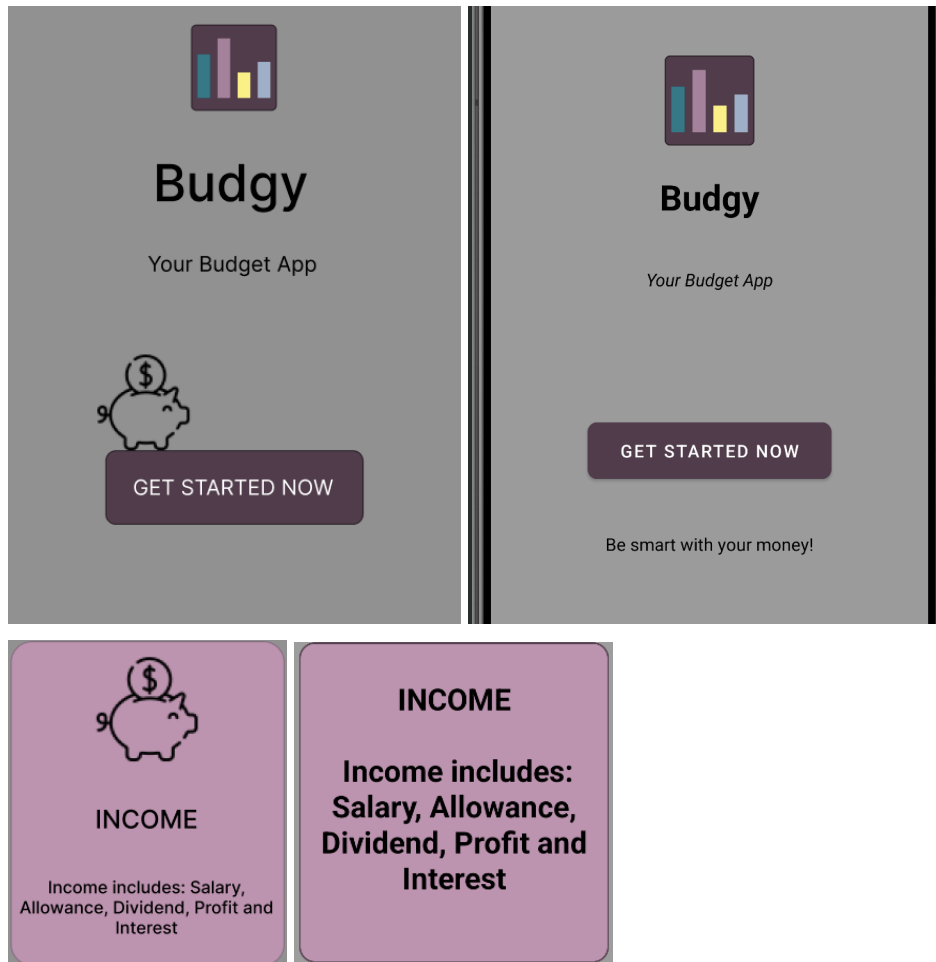
Innan (till vänster) - Slutgiltigt (till höger)



4. Ren design

- Valde att inte ha med spargrisen då jag kände att jag ville ha en lite “mognare” look på appen.

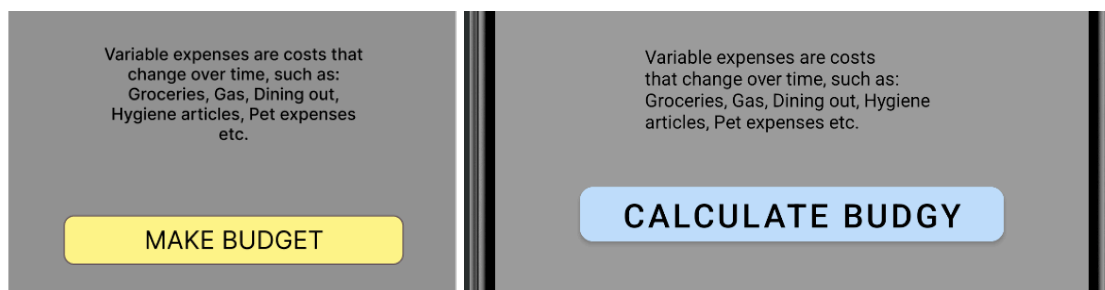
Design (till vänster) - Slutgiltigt (till höger)



5. Koppla tillbaka till App-namnet

- Valde att ändra knappen “Make Budget” till “Calculate Budgy” för att sluta cirkeln och koppla tillbaka till det catchiga namnet Budgy.

Design (till vänster) - Slutgiltigt (till höger)



Form

Budgy är en budgetapp skriven i Java för Android-plattformen. Den hjälper användaren att registrera inkomster och utgifter, kategorisera dem och få en visuell sammanfattning via ett cirkeldiagram. Appen består av flera aktiviteter (sidor) och klasser som hanterar inmatning, beräkning och presentation av data.

Grundläggande struktur (Java)

Java utgör grundstrukturen och funktionaliteten för appen. Applikationen består av följande **huvudkomponenter**:

- Start.java – Startskärmen där användaren initierar budgetprocessen.
- Income.java – Hanterar inmatning av inkomst.
- FixedExpense.java – Hanterar fasta utgifter.
- LoanAndCredit.java – Hanterar lån och krediter.
- VariableExpense.java – Hanterar rörliga utgifter.
- Summary.java – Visar en sammanfattning inklusive ett cirkeldiagram.
- Expense.java – En klass som lagrar och hanterar inkomster och utgifter.
- ExpenseList.java – En adapter för att visa utgifter i en RecyclerView (lista).

Exempel på grundläggande Java (Start.java):

```
Start.java x
1 package com.budgy;
2
3 > import ...
7
8 /**
9  * Start är den första aktiviteten i budgetappen.
10  * Den visar en startskärm med en knapp som låter användaren påbörja budgetprocessen.
11  */
12  annastrombeerg
13
14 public class Start extends AppCompatActivity {
15
16     /**
17      * Initialiserar aktiviteten och sätter layouten.
18      * Hämtar referens till startknappen och sätter en lyssnare för att navigera vidare.
19      *
20      * @param savedInstanceState Om aktiviteten återställs sparas tidigare tillstånd här.
21      */
22     annastrombeerg
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.start);
27
28         Button startButton = findViewById(R.id.start_button);
29         startButton.setOnClickListener(v -> {
30             //Navigera vidare till nästa
31             Intent intent = new Intent( packageContext: Start.this, Income.class);
32             startActivity(intent);
33         });
34     }
35 }
```

Stil och layout (XML)

Budgy använder sig av en modern och responsiv design med ConstraintLayout, anpassade färgteman och stilade UI-element.

- Huvudlayout (ConstraintLayout): Alla skärmar är uppbyggda med ConstraintLayout för flexibilitet och anpassning till olika skärmstorlekar.
- Teman och färgscheman: Budgy använder Material Design med ett färgtema där primärfärgen är lila och sekundärfärgen är gul för en tydlig och modern känsla (themes.xml).
- Runda UI-element: Vissa TextView, Button, EditText har rundade hörn (rounded_textview.xml, rounded_edittext.xml, rounded_button.xml & rounded_bottom.xml).
- Knappdesign: Knappstilar inkluderar NextButton och ToTheEndButton för konsekvent design.

- Cirkeldiagram: Budgetfördelningen visas med ett PieChart i Summary-aktiviteten.

Exempel på XML (start.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    package="com.budgy">

    <ImageView
        android:id="@+id/logo_image"
        android:layout_width="124dp"
        android:layout_height="92dp"
        android:src="@drawable/app_icon"
        app:layout_constraintBottom_toTopOf="@+id/start_button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.209"/>

    <TextView
        android:id="@+id/app_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="28dp"
        android:text="Budgy"
        android:textColor="@color/black"
        android:textSize="36sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/logo_image"/>

```

Viktiga kodavsnitt

Exempel på viktiga kodavsnitt nedan

Inkomsthantering (Income.java):

```
/**
 * Anropas efter att texten har ändrats.
 * Säkerställer att " KR" alltid läggs till i slutet av inmatningen och att markören placeras korrekt.
 */
@annastrombeerg
@Override
public void afterTextChanged(Editable editable) {
    if (isEditing) return; //Undviker rekursiv ändring av text
    isEditing = true;
    //Tar bort eventuell befintlig " KR" text och pensar mellanslag
    String text = editable.toString().replace(target: " KR", replacement: "").trim();
    if (!text.isEmpty()) { //Om texten inte är tom, lägg till " KR" och uppdatera texten i EditText
        text += " KR";
        income.setText(text);
        income.setSelection(text.length() - 3);
    }
    isEditing = false;
}
});
}
```

```

109      /**
110       * Sparar inkomsten från EditText till Expense-klassen.
111       */
112      2 usages  ↳ annastrombeerg
113      private void saveIncome() {
114          double incomeValue = 0;
115          if (!income.getText().toString().isEmpty()) {
116              incomeValue = Double.parseDouble(income.getText().toString().replace(target: " KR", replacement: "").trim());
117          }
118          Expense.setIncome(incomeValue);
119      }

```

Pseudokod:

Metod afterTextChanged (editable)

Om isEditing är true then

return

Slut

isEditing = true

text = ersätt(editable, " KR" med "")

text = trim(text)

Om text inte är tom then

text = text + " KR"

Sätt värde på income till text

Placera markören vid textlängd - 3

Slut

isEditing = false

Slut

Metod saveIncome

incomeValue = 0

Om income.getText() inte är tom then

incomeValue = konvertera till decimaltal

(ersätt(income.getText(), " KR" med ""), trim)

Slut

Anropa Expense-metod setIncome(sätt till incomeValue)

Slut

Hantering av utgifter (Expense.java):

```
66      /**
67       * Lägger till en fast utgift i listan över fasta utgifter.
68       *
69       * @param name Namnet på den fasta utgiften.
70       * @param amount Beloppet för den fasta utgiften.
71       */
72      1 usage  🧑 annastrombeerg
73      public static void addFixedExpense(String name, double amount) {
74          |   fixedExpenses.add(new Expense(name, amount));
75      }
76  ≡
77      /**
78       * Lägger till en låne- eller kreditutgift i listan över lån och krediter.
79       *
80       * @param name Namnet på lånet eller krediten.
81       * @param amount Beloppet för lånet eller krediten.
82       */
83      1 usage  🧑 annastrombeerg
84      public static void addLoanCredit(String name, double amount) {
85          |   loanCredits.add(new Expense(name, amount));
86      }
87
88      /**
89       * Lägger till en rörlig utgift i listan över rörliga utgifter.
90       *
91       * @param name Namnet på den rörliga utgiften.
92       * @param amount Beloppet för den rörliga utgiften.
93       */
94      1 usage  🧑 annastrombeerg
95      public static void addVariableExpense(String name, double amount) {
96          |   variableExpenses.add(new Expense(name, amount));
97      }
```



```

150      /**
151       * Återställer alla data i Expense-klassen, inklusive inkomster och utgifter.
152       */
153      1 usage  📄 annastrombeerg
154      public static void resetData() {
155          income = 0;
156          fixedExpenses.clear();
157          loanCredits.clear();
158          variableExpenses.clear();
159      }
160
161      /**
162       * Tar bort en fast utgift från listan baserat på dess position.
163       *
164       * @param position Indexet för den utgift som ska tas bort.
165       */
166      1 usage  📄 annastrombeerg
167      public static void removeFixedExpense(int position) {
168          if (position >= 0 && position < fixedExpenses.size()) {
169              fixedExpenses.remove(position);
170          }
171      }

```

Note: remove-metoden finns även för LoanAndCredit-/VariableExpense.java

Pseudokod:

```

Metod addFixedExpense (name och amount)
    Skapa nytt Expense objekt med (name och amount)
    Lägg till objektet i fixedExpenses-listan
Slut

Metod addLoanCredit (name och amount)
    Skapa nytt Expense objekt med (name och amount)
    Lägg till objektet i loanCredits-listan
Slut

Metod addVariableExpense (name och amount)
    Skapa nytt Expense objekt med (name och amount)
    Lägg till objektet i variableExpenses-listan
Slut

```

```

Metod resetData
    income = 0
    Rensa fixedExpenses-listan
    Rensa loanCredits-listan

```

```

Rensa variableExpenses-listan
Slut

Metod removeFixedExpense (position)
    Om position >= 0 och position < storleken av
fixedExpenses-listan then
        Ta bort objekt vid position från fixedExpenses-listan
    Slut
Slut

```

Hantering av fasta utgifter (FixedExpense.java - finns samma i
LoanAndCredit-/VariableExpense.java):

```

52      /**
53       * Hanterar klick på "Next"-knappen.
54       * Lägger till en ny fast utgift i listan.
55       * Om både namn och belopp är angivna, läggs utgiften till i Expense-klassen
56       * och listan uppdateras i RecyclerView.
57       */
58      addExpense.setOnClickListener(v -> {
59          String name = expenseName.getText().toString().trim();
60          String amount = expenseAmount.getText().toString().trim();
61
62          if (!name.isEmpty() && !amount.isEmpty()) {
63              double addAmount = Double.parseDouble(amount);
64              Expense.addFixedExpense(name, addAmount);
65
66              //Uppdatera listan
67              expenses.clear();
68              expenses.addAll(Expense.getFixedExpenses());
69              adapter.notifyDataSetChanged();
70
71              //Rensa inputfält och uppdatera totalsumman
72              expenseName.setText("");
73              expenseAmount.setText("");
74              updateTotalExpense();
75          }
76      });

```

```

91      /**
92       * Tar bort en utgift från listan baserat på angiven position.
93       * Efter borttagning uppdateras listan i RecyclerView och den totala summan beräknas om.
94       *
95       * @param position Positionen för utgiften som ska tas bort.
96       */
97      1 usage  🡕 annastrombeerg
98      @Override
99      public void onDeleteExpense(int position) {
100
101          Expense.removeFixedExpense(position);
102
103          //Uppdatera listan efter borttagning
104          expenses.clear();
105          expenses.addAll(Expense.getFixedExpenses());
106          adapter.notifyDataSetChanged();
107          updateTotalExpense();
108      }
109
110      /**
111       * Uppdaterar TextView som visar den totala summan av fasta utgifter.
112       */
113      3 usages  🡕 annastrombeerg
114      private void updateTotalExpense() {
115          totalExpense.setText("TOTAL: " + Expense.getTotalFixedExpenses() + " KR");
116      }
117  }

```

Pseudokod:

Metod addExpense setOnClickListener

När användaren klickar på addExpense then

name = hämta text från expenseName och trimma

amount = hämta text från expenseAmount och trimma

Om name inte är tom och amount inte är tom then

addAmount = konvertera amount till decimaltal

Anropa Expense-metod addFixedExpense(lägg till objekt med name och addAmount)

Rensa expenses-listan

Lägg till alla objekt från Expense.getFixedExpenses() i expenses-listan

Uppdatera adapter med notifyDataSetChanged()

Sätt expenseName till tomt

Sätt expenseAmount till tomt

Anropa updateTotalExpense()

Slut

Slut

Slut

Metod onDeleteExpense (position)

Anropa Expense-metod removeFixedExpense(ta bort objekt vid position)

Rensa expenses-listan

Lägg till alla objekt från Expense.getFixedExpenses() i expenses-listan

Uppdatera adapter med notifyDataSetChanged()

Anropa updateTotalExpense()

Slut

Metod updateTotalExpense

Sätt texten i totalExpense till "TOTAL: " + Expense.getTotalFixedExpenses() + " KR"

Slut

Sammanfattning med cirkeldiagram (Summary.java):

```
77  /**
78   * Beräknar och visar PieChart med fördelning av inkomster och utgifter.
79   * Hämtar de aktuella budgetvärdena och beräknar procentandelar för varje kategori.
80   * Uppdaterar PieChart och TextViews med procentandelen av varje kategori.
81   *
82   * @param pieChart PieChart som ska uppdateras med budgetfördelning.
83   */
84  @usage 1 annastromberg
85  private void setupPieChart(PieChart pieChart) {
86      //Hämta totala inkomst och utgifter från Expense.java
87      double totalIncome = Expense.getTotalIncome();
88      double totalFixed = Expense.getTotalFixedExpenses();
89      double totalLoan = Expense.getTotalLoanCredits();
90      double totalVariable = Expense.getTotalVariableExpenses();
91      double totalBudget = totalIncome + totalFixed + totalLoan + totalVariable;
92
93      //Hämta TextViews från layouten
94      TextView incomePercentage = findViewById(R.id.income_percentage);
95      TextView fixedPercentage = findViewById(R.id.fixed_percentage);
96      TextView loanPercentage = findViewById(R.id.loanncred_percentage);
97      TextView variablePercentage = findViewById(R.id.variable_percentage);
98
99      //Rensar PieChart för att ta bort tidigare data innan nya värden läggs till
100     pieChart.clearChart();
101
102     /**
103      * Beräknar procentandelen av varje kategori i budgeten och uppdaterar TextViews samt PieChart.
104      * Procentvärdena visas bredvid varje kategori och PieChart-sektionerna skapas dynamiskt.
105      */
106     if (totalBudget > 0) {
107         float incomePercent = (float) ((totalIncome / totalBudget) * 100);
108         float fixedPercent = (float) ((totalFixed / totalBudget) * 100);
109         float loanPercent = (float) ((totalLoan / totalBudget) * 100);
110         float variablePercent = (float) ((totalVariable / totalBudget) * 100);
111
112         //Uppdatera TextViews med procent
113         incomePercentage.setText("Income: " + String.format("%.1f", incomePercent) + "%");
114         fixedPercentage.setText("Fixed: " + String.format("%.1f", fixedPercent) + "%");
115         loanPercentage.setText("Loan/Credit: " + String.format("%.1f", loanPercent) + "%");
116         variablePercentage.setText("Variable: " + String.format("%.1f", variablePercent) + "%");
117
118         //Lägg till "slices" i PieChart
119         if (totalIncome > 0)
120             pieChart.addPieSlice(new PieModel( "legendLabel: \"Income\", (float) totalIncome, Color.parseColor( colorString: \"#B8E1FF\")));
121         if (totalFixed > 0)
122             pieChart.addPieSlice(new PieModel( "legendLabel: \"Fixed\", (float) totalFixed, Color.parseColor( colorString: \"#087E8B\")));
123         if (totalLoan > 0)
124             pieChart.addPieSlice(new PieModel( "legendLabel: \"Loan/Credit\", (float) totalLoan, Color.parseColor( colorString: \"#FF272F\")));
125         if (totalVariable > 0)
126             pieChart.addPieSlice(new PieModel( "legendLabel: \"Variable\", (float) totalVariable, Color.parseColor( colorString: \"#C492B1\")));
127     }
128     //Slutligen startas en animation för PieChart.
129     pieChart.startAnimation();
130 }
131 }
```

Pseudokod:

Metod setupPieChart (pieChart)

totalIncome = hämta från Expense.getTotalIncome()

totalFixed = hämta från Expense.getTotalFixedExpenses()

```

totalLoan = hämta från Expense.getTotalLoanCredits()
totalVariable = hämta från Expense.getTotalVariableExpenses()
Sätt totalBudget = totalIncome + totalFixed + totalLoan +
totalVariable

incomePercentage = hämta TextView med id: income_percentage
fixedPercentage = hämta TextView med id: fixed_percentage
loanPercentage = hämta TextView med id: loanncred_percentage
variablePercentage = hämta TextView med id:
variable_percentage

Rensa pieChart

Om totalBudget > 0 then
    Sätt incomePercent = (totalIncome / totalBudget) * 100
    Sätt fixedPercent = (totalFixed / totalBudget) * 100
    Sätt loanPercent = (totalLoan / totalBudget) * 100
    Sätt variablePercent = (totalVariable / totalBudget) * 100

    Sätt incomePercentage text till "Income: " + incomePercent
+ "%"
    Sätt fixedPercentage text till "Fixed: " + fixedPercent +
 "%"
    Sätt loanPercentage text till "Loan/Credit: " +
loanPercent + "%"
    Sätt variablePercentage text till "Variable: " +
variablePercent + "%"

    Om totalIncome > 0 then
        Lägg till PieSlice med namn "Income" + totalIncome och
färg "#B8E1FF"
    Om totalFixed > 0 then
        Lägg till PieSlice med namn "Fixed" + totalFixed och
färg "#087E8B"
    Om totalLoan > 0 then
        Lägg till PieSlice med namn "Loan/Credit" + totalLoan
och färg "#FFF275"
    Om totalVariable > 0 then
        Lägg till PieSlice med namn "Variable" + totalVariable
och färg "#C492B1"
    Slut

```

```
    Starta piechart animation
  Slut
```

Helheten vid vanligt use case

Pseudokod:

Program Start

Visa startskärm

Om användare trycker på "Get Started Now" → Gå till Income
Slut

Program Income

Visa inkomstfält

Vänta på inmatning

Om "Next" trycks → Spara inkomst → Gå till FixedExpense
Slut

Program FixedExpense

Visa namn- och beloppfält

Vänta på inmatning

Om "Add" trycks → Spara utgift och lägg till i listan

Visa lista över fasta utgifter

Om "Soptunna" trycks → Radera utgift och uppdatera listan

Om "Next" trycks → Spara utgift/utgifter → Gå till

LoanAndCredit

Slut

Program LoanAndCredit

Visa namn- och beloppfält

Vänta på inmatning

Om "Add" trycks → Spara utgift och lägg till i listan

Visa lista över lån och krediter

Om "Soptunna" trycks → Radera utgift och uppdatera listan

Om "Next" trycks → Spara utgift/utgifter → Gå till

VariableExpense

Slut

Program VariableExpense

Visa namn- och beloppfält

Vänta på inmatning

Om "Add" trycks → Spara utgift och lägg till i listan

Visa lista över rörliga utgifter

Om "Soptunna" trycks → Radera utgift och uppdatera listan

Om "Calculate" trycks → Spara utgift/utgifter → Gå till

Summary

Slut

Program Summary

Beräkna **total** inkomst och utgifter

Visa PieChart

Visa balance efter uträkning

Om "**Start Over**" trycks → **Gå** till Start och nollställ allt

Om "**Change Income**" trycks → **Gå** till Income

Om "**Change Expenses**" trycks → **Gå** till FixedExpense

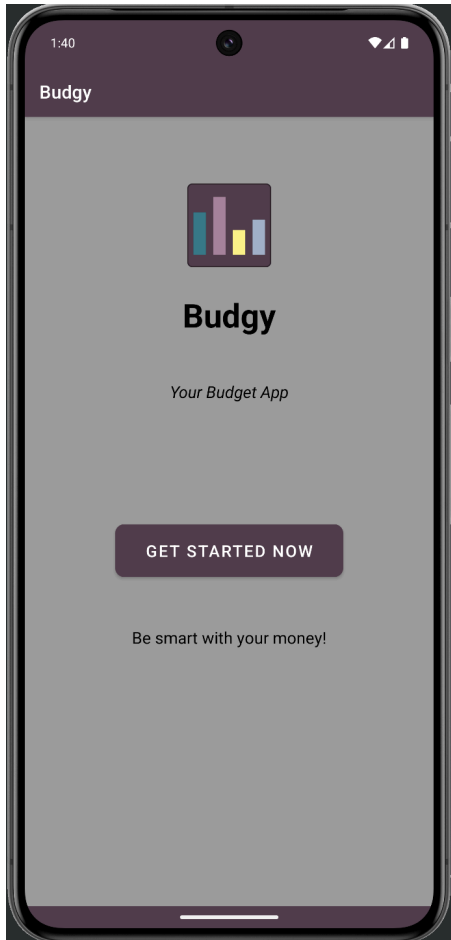
Slut

Funktion

Användarens första interaktion

När användaren startar appen visas en välkomstkärm med appens namn och en startknapp. När knappen trycks ned navigeras användaren till inkomstinmatningen.

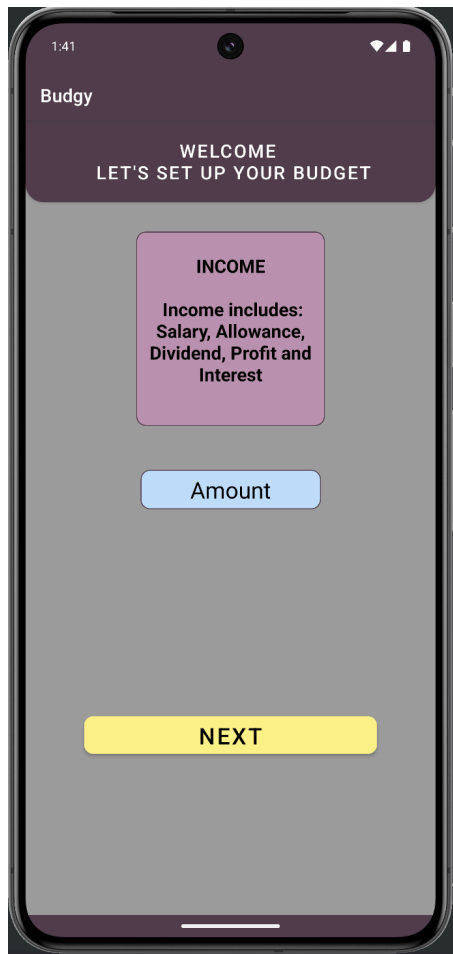
Startskärm/välkomstkärm



Inkomstinmatning

Användaren uppmanas att ange sin månatliga inkomst. Ett numeriskt inmatningsfält säkerställer att endast siffror kan anges. När användaren trycker på "Next" sparas inkomsten och användaren skickas vidare till nästa skärm.

Inkomstinmatningsskärm



Hantering av utgifter

Efter inkomstinmatningen går användaren igenom tre olika kategorier av utgifter:

1. **Fasta utgifter** (t.ex. hyra, försäkring)
2. **Lån och krediter** (t.ex. billån, kreditkortsskulder)
3. **Rörliga utgifter** (t.ex. mat, nöje)

Varje kategori har ett formulär där användaren kan namnge utgiften och ange belopp. En lista visar de tillagda utgifterna och en total summa uppdateras dynamiskt.

Fasta utgifter - skärm (fullständig lista)

1:43

Budgy

ADD FIXED EXPENSES

Expense Name

Amount

ADD

Rent	10500.0 KR	🗑️
Insurance	200.0 KR	🗑️
Phone	200.0 KR	🗑️
Gym	500.0 KR	🗑️

TOTAL: 11400.0 KR

Fixed expenses are costs in your budget that do not vary from month to month, such as: Rent, Insurance, Membership/Subscription costs, Phone bill etc.

NEXT

Fasta utgifter - skärm (fullständig lista EFTER borttagning)

1:47

Budgy

ADD FIXED EXPENSES

Expense Name

Amount

ADD

Rent	10500.0 KR	
Insurance	200.0 KR	

TOTAL: 10700.0 KR

Fixed expenses are costs in your budget that do not vary from month to month, such as: Rent, Insurance, Membership/Subscription costs, Phone bill etc.

NEXT

Lån och krediter - skärm

The screenshot shows the 'Lån och krediter' (Loans and Credits) screen in the Budgy app. The screen has a dark purple header with the time '1:44' and status icons. Below the header, the app name 'Budgy' is visible. The main section is titled 'ADD LOANS AND CREDITS' and contains two input fields: 'Expense Name' and 'Amount'. Below these fields is a yellow 'ADD' button. A list of existing entries is shown below the 'ADD' button, with two items: 'Student Loan' (789.0 KR) and 'Car Loan' (2995.0 KR). Each item has a trash icon to its right. Below the list, the total amount is displayed as 'TOTAL: 3784.0 KR'. A note below the total states: 'Loan and credits include: Car loans, Personal loans, Student loans, Credit bills and loans etc.' At the bottom of the screen is a yellow 'NEXT' button.

Expense Name	Amount	Action
Student Loan	789.0 KR	
Car Loan	2995.0 KR	

TOTAL: 3784.0 KR

Loan and credits include: Car loans, Personal loans, Student loans, Credit bills and loans etc.

Rörliga utgifter - skärm

1:45

Budgy

ADD VARIABLE EXPENSES

Expense Name

Amount

ADD

Gas	1000.0 KR	🗑️
Groceries	3000.0 KR	🗑️

TOTAL: 4000.0 KR

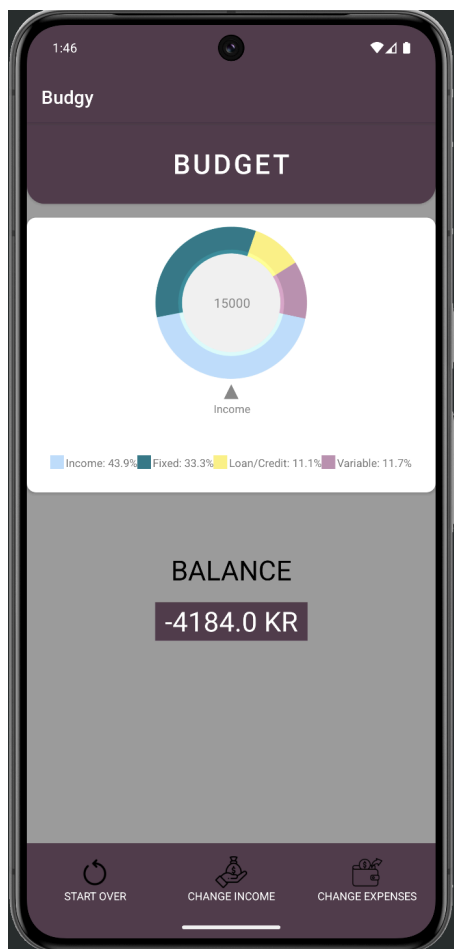
Variable expenses are costs that change over time, such as: Groceries, Gas, Dining out, Hygiene articles, Pet expenses etc.

CALCULATE BUDGY

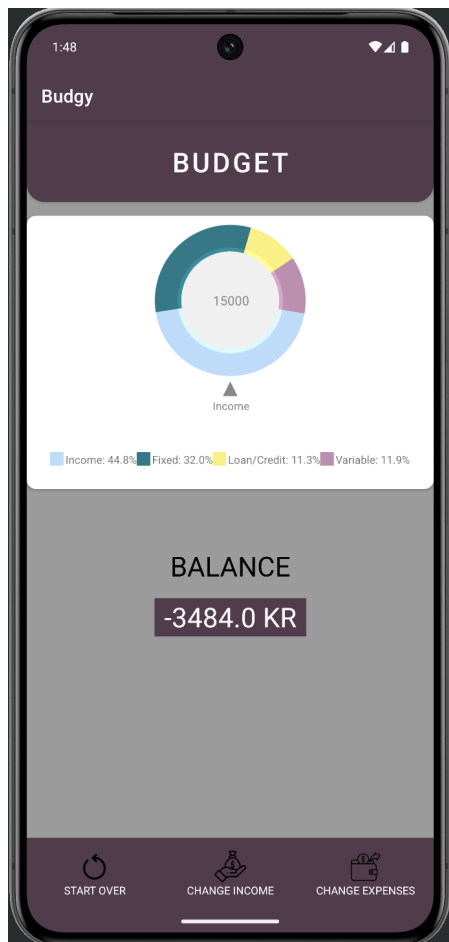
Sammanfattning och analys

När alla utgifter har registrerats visas en sammanfattningsskärm med en översikt av användarens ekonomi. Ett cirkeldiagram visualiserar fördelningen mellan inkomst och olika typer av utgifter. Balansen (inkomst minus utgifter) visas tydligt och användaren kan gå tillbaka och ändra information om det behövs.

Sammanfattningsskärm



Sammanfattningsskärm (efter borttagning av de fasta utgifterna tidigare)

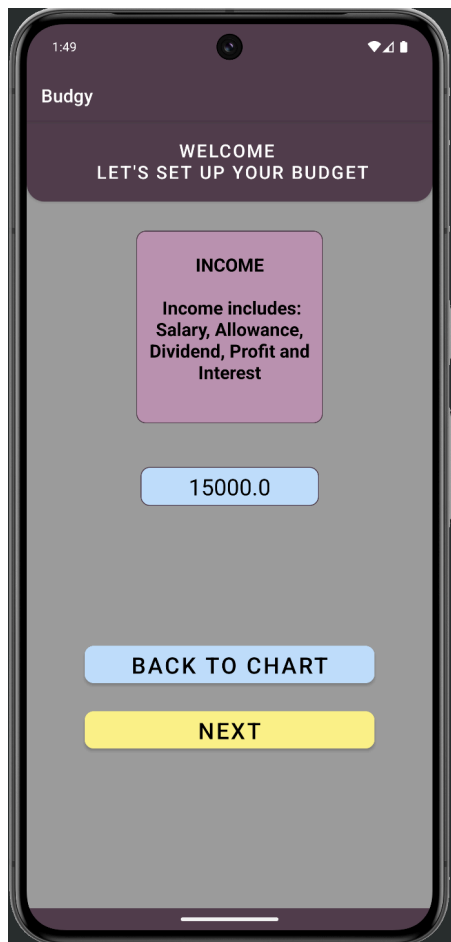


Navigering och interaktion

- Tillbaka-knappar (Start Over, Change Income & Change Expenses) låter användaren navigera tillbaka till tidigare skärmar för att redigera information
- Knaptryck och visuell feedback skapar en intuitiv upplevelse
- Alla numeriska fält formateras automatiskt för att säkerställa korrekt inmatning

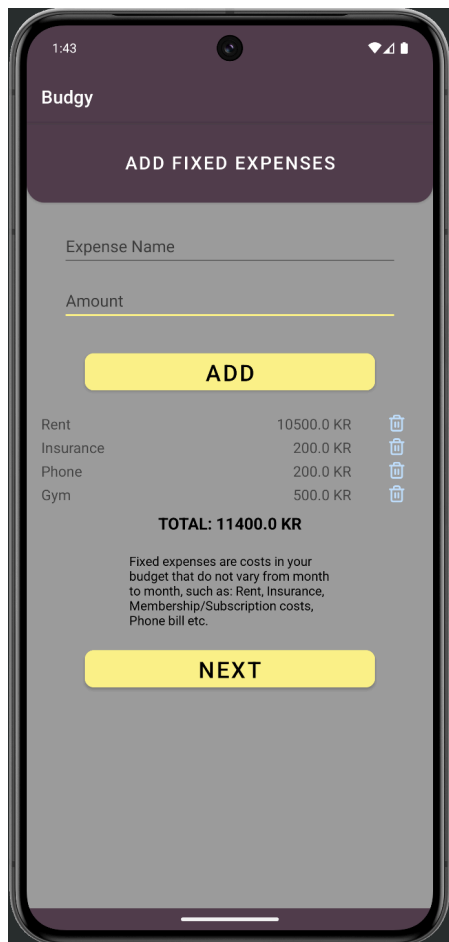
När användaren klickar på "Start Over" så nollställs allt och användaren är tillbaka på start/välkomstskärmen.

När användaren klickar på "Change Income" så kommer användaren tillbaka till inkomstinmatningsskärmen där inkomsten som först matades in fortfarande finns kvar:



Och nu finns en knapp "BACK TO CHART" som tar användaren direkt tillbaka till Piecharten efter att inkomst ändrats.

När användaren klickar på "Change Expenses" så kommer användaren tillbaka till skärmen för fasta utgifter:



Där ligger alla utgifter användaren tidigare matade in kvar. Användaren kan då ta bort eller lägga till utgifter. För att komma vidare går användaren genom lån och krediter, samt rörliga utgifter för att sedan "Calculate" sammanfattningen igen.