***Технически университет – София***

*Факултет по компютърни системи и технологии*

*Специалност - КСИ*

*КУРСОВ ПРОЕКТ*

*по дисциплината : Програмиране за мобилни устройства*

*тема : Приложение за домашно счетоводство.*

*изготвен от : преподавател :*

*Ангел Йорданов гл. ас. Невен Николов*

*(III курс, 45.гр. фак. №: 121220080)  
 Георги Янушев*

*(III курс, 45.гр. фак. №: 121220155)  
 Катерина Чамова*

*(III курс, 45.гр. фак. №: 121220179)*

# **УВОД**

Задачата да управлявате своите финанси в домашния си бюджет може да бъде предизвикателна, но с правилните инструменти и подход, можете да го направите по-лесно и по-ефективно. Ние знаем, че управлението на личните финанси може да бъде трудно и изисква много време и усилия, затова сме разработили това приложение, за да Ви помогнем да следите по-лесно и ефективно разходите и приходите си.

Нашето приложение Ви дава възможност да създавате бюджети, да следите разходите си в реално време, да анализирате своята финансова ситуация и да получавате уведомления, когато сте в дълг. Това Ви помага да контролирате своите разходи и да управлявате по-добре своята финансова ситуация. Надяваме се, че нашето приложение ще Ви помогне да управлявате по-добре своите финанси и да постигнете финансова стабилност.

# **Анализ на съществуващи разработки**

## **1.** **Mint**

* 1. Апликацията Mint има сериозен гръб в лицето на компанията майка Intuit, която също така стои зад големите брандове QuickBooks и TurboTax. Mint позволява създаването на бюджет, следене на разходите, както и някои по-иновативни функции като свързване на банковите сметки към апликацията и регулярно следене на месечните сметки.
  2. Апликацията позволява на потребителите да създават нотификации за предстоящи плащания, да се изпращат предупреждение, ако стигнем определен лимит на месечните ни бюджет и също така създава напълно безплатно кредитна оценка.

## **2.** **iSaveMoney**

* 1. iSaveMoney е друг вариант за стандартно приложение за управление на лични финанси. Приложението позволява да изготвяме бюджет и да записваме разходите си по различни предварително дефинирани категории. Приложението позволява и да извличаме данните за последващ анализ чрез Excel, например, както и да правим визуализации на нашите приходи, разходи, спестени пари и др.
  2. Приложението има безплатна версия с ограничение на функционалностите както и платена версия с еднократна сравнително ниска такса.

## **3.** **Spending Tracker**

* 1. Spending Tracker е приложение, чиято цел е да ни помогне да пестим пари чрез контролиране на разходите ни. Целта на приложението е да записваме всеки приход и разход през месеца. По отношение на разноските категоризацията е важна, защото в края на месеца имаме възможност да видим таблично и графично за какво харчим парите си.
  2. Платената версия на приложението дава достъп до допълнителни функционалности като записването на повтарящи се разходи.

## **4.** **Wally**

* 1. Wally е напълно безплатно приложение, което специализира в следенето на разходите. Приложението позволява да се записват разходите чрез заснемане на касовите бележки, което записва не само сумата, но и локацията на направения разходи. Това спестява време, а функционалността „докладване“ позволява да направим един по-обстоен преглед къде отиват парите ни всеки месец, тримесечие или дори година.

# **Проектиране**

## **1.** **Проектиране на структурата на приложение за домашно счетоводство.**

Приложението е реализирано чрез програмния език Java. Java е обектно-ориентиран език за програмиране на високо ниво, базиран на класове, който е проектиран да има възможно най-малко зависимости при изпълнението. Това е език за програмиране с общо предназначение, предназначен да позволи на програмистите да пишат веднъж, да се изпълняват навсякъде (WORA) което означава, че компилиране Java код може да работи на всички платформи, които поддържат Java, без да е необходимо повторно компилиране. Java приложенията обикновено се компилират в байт код, който може да работи на всяка Java виртуална машина (JVM), независимо от основната компютърна архитектура. Синтаксисът на Java е подобен на C и C++, но има по-малко средства на ниско ниво от всеки от тях. Средата за изпълнение на Java предоставя динамични възможности (като отражение и модификация на кода по време на изпълнение), които обикновено не са налични в традиционните компилирани езици. Към 2019 г. Java е един от най-популярните използвани езици за програмиране според GitHub особено за клиент-сървър уеб приложения, с докладвани 9 милиона разработчици.

1.1. Създадени функционалности.

1.1.1. Навигиране между екраните на приложението чрез използване на Intent в дадените Activity-а.

1.1.2. Връзка между Layout(front end) и Activity(back end)

1.1.3. Реализиране на персонализирани Adapter-и с цел връзка между Listview-та и персонализирани Layout-ти.

1.1.4. Използване на Pop Up прозорци чрез използването на AlertDialog.Builder.

1.1.4.1. Изскачащите прозорци в приложението ни помагат в улесняване и опростяване на Ui-ат чрез свързвайки ги с действия които обикновено биха довели до навигацията до друг екран или нуждата от голям брой бутони на сегашния екран.

1.1.5. Използване на известия на мобилното устройство.

1.1.5.1. Използването на известия ни помага да напомняме на потребителя в случай че той има не платен заем/дълг.

**2.** **Класове**

2.1. Activities

2.1.1. DebtsActivity- свързано е със и съдържа бизнес логиката на екранът debts\_screen. DebtsActivity-то служи за създаването манипулирането и прегледа на дългове, представени от класът Debts.

2.1.2. HomeActivity- свързано е със и съдържа бизнес логиката на екранът home\_screen. HomeActivity-то служи за навигирането между останалите екрани и служи за прегледа на текущия баланс на потребителя.

2.1.3. SavingsActivity- свързано е със и съдържа бизнес логиката на екранът savings\_screen. SavingsActivity-то служи за създаването на спестителни фондове с цел по лесен начин за следене на спестяванията на потре- бителя. Допълнително служи за манипулирането на тези спестявания както и техния преглед. Спестявания- та се представени от класът Savings.

2.1.4. SchedulingActivity- свързано е със и съдържа бизнес логиката на екранът scheduling\_screen. SchedulingActivity-то служи за създаването на автоматични планове за въвеждане на приходи които се изпълняват периодично, пример заплата.

Допълнително служи за прекратяването на тези планове, както и техния преглед. Плановете са представени от класът Scheduling.

2.1.5. TransactionActivity- свързано е със и съдържа бизнес логиката на екранът transactio\_screen. TransactionActivity-то служи за създаването транзакции, от тип приход или разход. Транзакциите са представени от класът Transactions

2.1.6. TransactionHistoryActivity- свързано е със и съдържа бизнес логиката на екранът transaction\_history. TransactionHistoryActivity-то служи за прегледат и катаголизирането на транзакции в три групи: седмични, месечни и годишни.

2.2. Adapters

2.2.1. DebtsListAdapter- адаптерът служи като връзка между custom\_list и debts\_listview намиращо се в debts\_screen с цел създаването на персонали- зиран дизайн за редът на listview-то.

2.2.2. SavingsListAdapter- адаптерът служи като връзка между savings\_list и savings\_listview намиращо се в savings\_screen с цел създаването на персонализиран дизайн за редът на listview-то.

2.2.3.SchedulingListAdapter- адаптерът служи като връзка между custom\_list и scheduling\_listview намиращо се в scheduling\_screen с цел създаването на персонализиран дизайн за редът на listview-то.

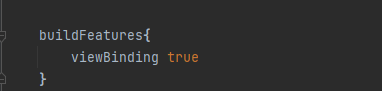
2.2.4.TransactionListAdapter- адаптерът служи като връзка между transaction\_list и listview намиращо се в transaction\_history с цел създаването на персонализиран дизайн за редът на listview-то.

2.3 Notifications

2.3.1 MyNotificationPublisher- клас който се изболзва от scheduleNotification, с цел публикуването на известия.

**Реализация**

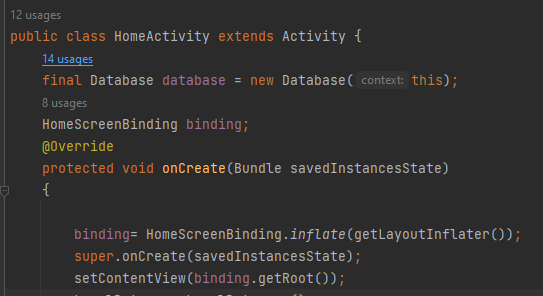
1. Bindings- връзката между Activity и екранът за който той отговаря е постигната с няколко стъпки
   1. Включването на viewBinding чрез build.gradle файлът. В файлът build.gradle добавяме следният код(фиг 1.1)



*фиг 1.1 Активиране на viewBinding*

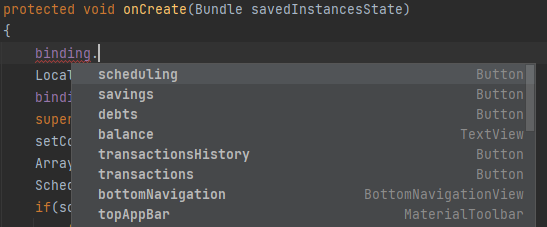
след въвеждането на кодът проектът трябва да се sync-не

* 1. След като е активиран автоматично ще бъдат добавени ScreenBinding класове отговарящи за всеки layout. При което в даденото Activity трябва да се инициализира променлива от избраният ScreenBinding клас (фиг 1.2)



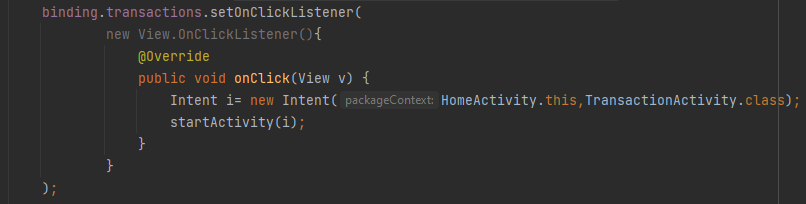
*фиг 1.2 инициализиране на ScreenBinding*

след като е променливата е инициализирана е нужно да setContentView-то да се промени в setContentView(binding.getRoot()); за да може при включването на Activity-то да се появява точният екран.

* 1. След това чрез binding. е възможно да се достъпи и манипулира всеки елемент от съответващия екран(фиг 1.3)

*фиг 1.3 демонстрация с HomeScreenBinding*

1. Навигация между екрани- реализирана чрез viewBinding за достъп до бутоните на Екранът и setOnClickListener-и и Intent
   1. Достигаме до избраният от нас бутон чрез променливата binding(фиг 2.1)

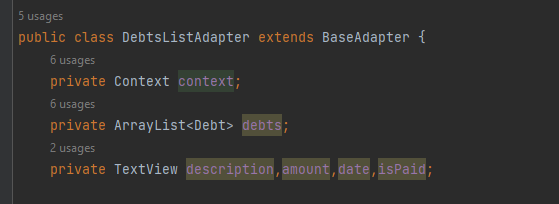


*фиг 2.1 Навигация между HomeActivity и TransactionActiity*

Нужно е използването на Intent в който се задават сегашното Activity и това към което искате да отидете, след което Intent-ът се извиква чрез startActivity()

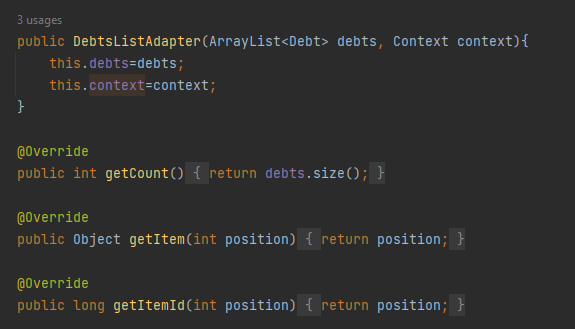
Intent- обект който се използва за комуникация между два компонента

1. Адаптери- реализират се с класове които наследяват класът BaseAdapter
   1. Създаваме променливи за context, ArrayList от нужният ни клас и същият брой Textview-та като custom\_list-а който искаме да използваме(фиг 3.1)



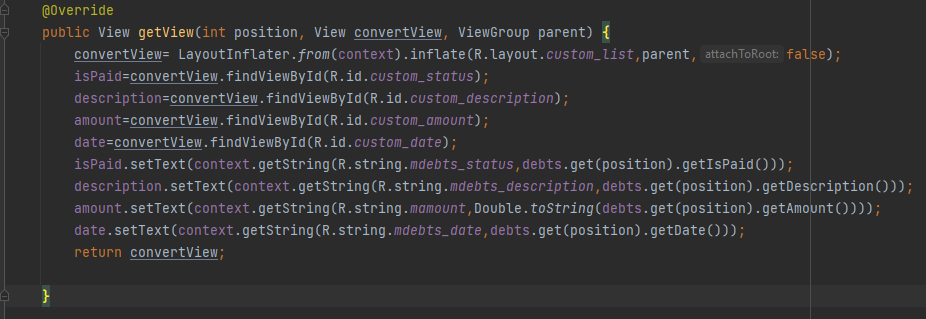
фиг 3.1 от DebtsListAdapter

* 1. Конструктор и имплементиране на нужните функции от BaseAdapter(фиг 3.2)



*фиг 3.2 от DebtsListAdapter*

* 1. Overriding GetView() (фиг 3.3)



*фиг 3.3*

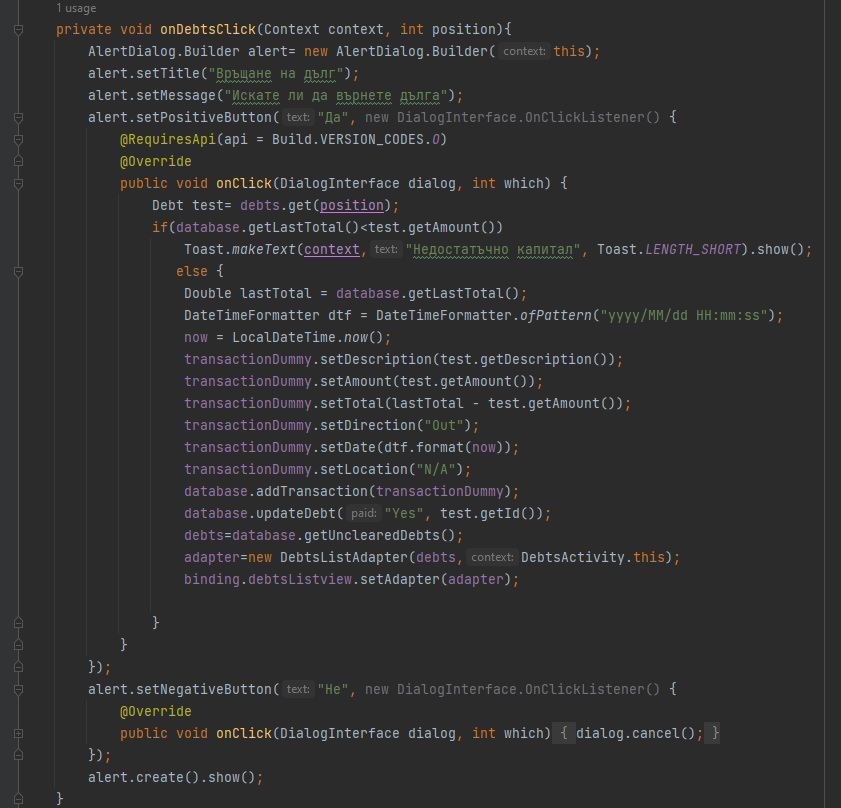
convertView приема layout-тът който сме избрали, custom\_list в този случай.

*Textview-тата приемат Textview-тата от custom\_list*

*Задаваме им желаните стойности, в този случай с стойности от ArrayList-та ни.*

*getString се ползва за взимане на вече създадени string-ове от values strings*

1. Pop Ups- реализират се чрез AlertDialog.Builder(фиг 4)



*фиг 4 onDebtsClick, функзия за създаване на pop up*

Инициализираме alert с контекст по избор

setTitle-за задаване на заглавие на прозорецът

setMessage- за задаване на текст на просорецът

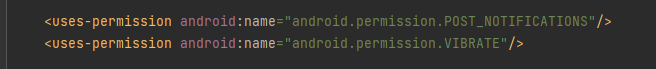
setPositiive за задаване текст на позитивният бутоните

setNegative за задаване текст на негативният бутоните

onClick какво да върши даденият бутон, в нашият случай служи за изплажане на дълг

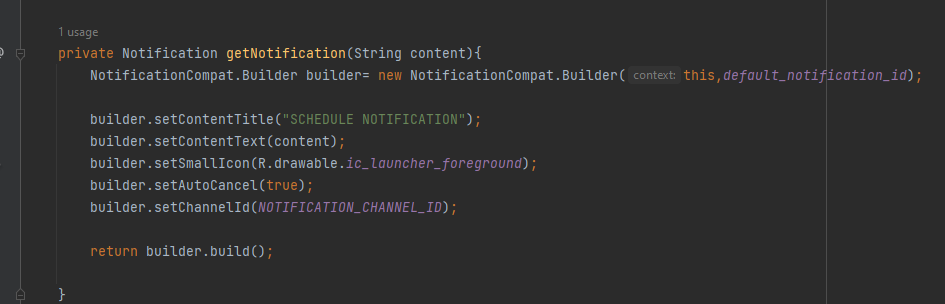
dialog.cancel затваря прозореца

1. Notifications- реализира се с помощта на няколко елемента
   1. Искане на позволение за изпращане на нотификации и вибрация на телефона(фиг 5.1)



*фиг 5.1 искане на позволение, случва се в AndroidManifest*

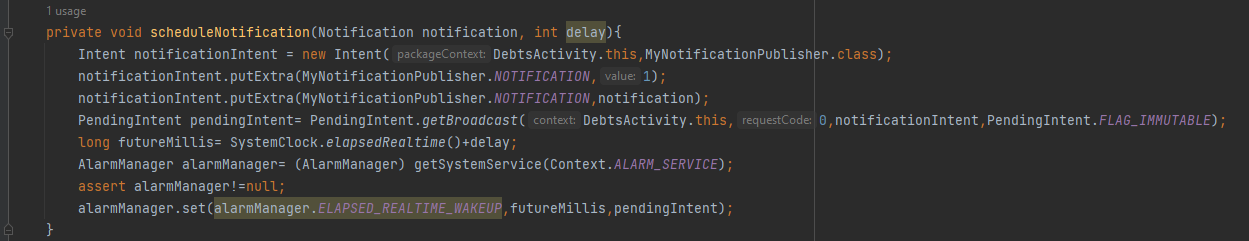
* 1. Метод за създаване на нотификация(фиг 5.2)



фиг 5.2 getNotification(), content е за задаване на текст за известието

При създаване на известие задаването на икона, заглавие и текст е задължително

* 1. Метод за планиране на нотификация(фиг 5.3)

фиг 5.3 scheduleNotification()

в него се извиква getNotification, delay се използва за да се зададе след колко време да се появи известието

* 1. MyNotificationPublisher(фиг 5.4)

фиг 5.4 класът наследява BroadcastReceiver

при API-та над 26та версия за показването на нотификация е нужен и канал на нотификациите, една от задачите на класът е да проверява за това и да създава канал ако е нужно

# 

# 

# 

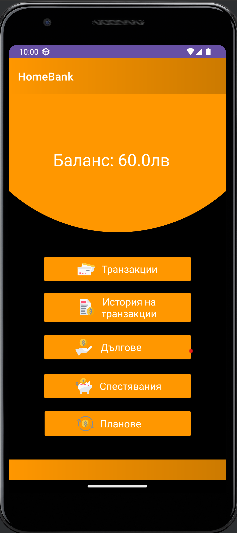
# 

# 

# 

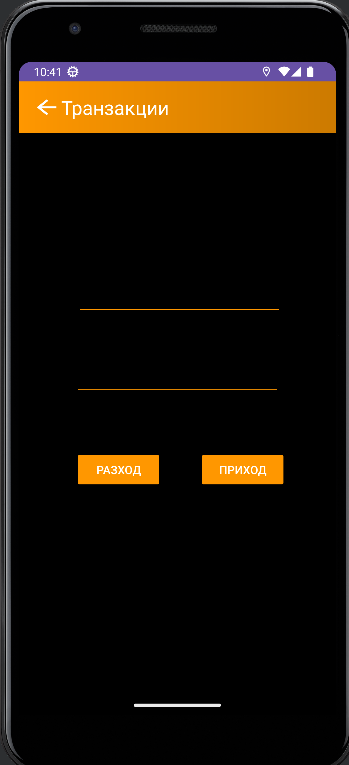
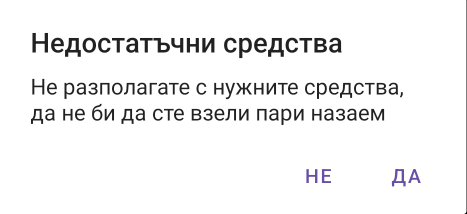
# Ръководство за потребителя

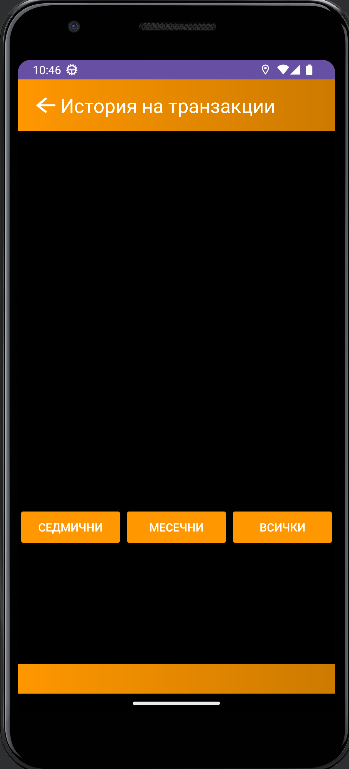
При отваряне на приложението потребителя вижда пред себе си основния екран:



Екрана показва част от възможностите на приложението и баланса на потребителя.

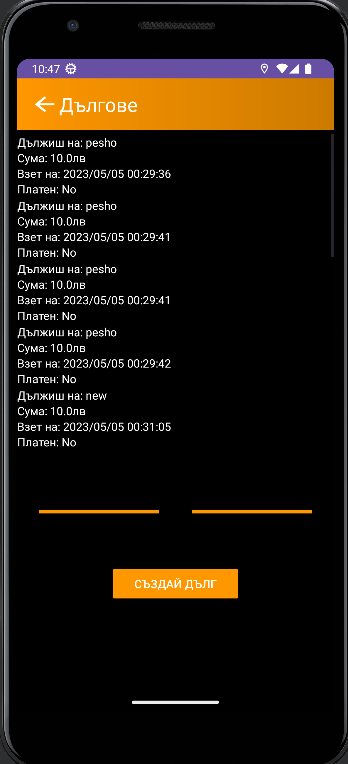
Има няколко бутона:

* Транзакции - след натискането на този бутон потребителя се отвежда на нов екран служещ за създаване на транзакция:  
    
  Следват възможностите за създаване на транзакция.
  + Първия вариант е да се добавят пари към текущия баланс на потребителя чрез попълване на полетата отговарящи за описание на транзакцията и за сумата, която ще се преведе и натискане на бутона “Приход”.
  + При желание на потребителя да направи транзакция за плащане трябва да се въведат отново описание и сума на транзакцията. При недостатъчен баланс се появява следното съобщение:  
      
    то има за цел да се обърне към потребителя за да разбере дали все пак физически разполагате със желаната сума.
    - При избор на възможност “Не” транзакцията се прекратява.
    - При избор на възможност “Да” води до екрана отговарящ за дълговете.
  + При избиране на стрелката в горния ляв ъгъл потребителя се отвежда до основното меню на приложението.
* История на транзакциите при избора на този бутон пред потребителя се разкрива следния екран:

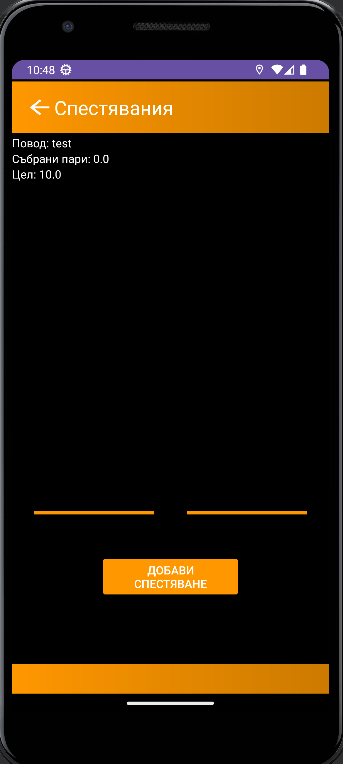


Откриват се следните възможности:

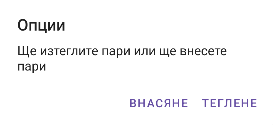
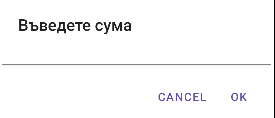
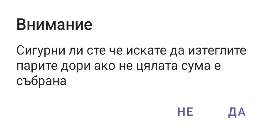
* + При натискане на бутона “Седмични” се появяват всички транзакции през изминалите 7 дни.
  + При натискане на бутона “Месечни” се показват пред потребителя всички транзакции направени през текущия месец
  + При избора “Всички” се показват всички транзакции по-стари от текущия месец
  + При избиране на стрелката в горния ляв ъгъл потребителя се отвежда до основното меню на приложението.
* Дългове екрана разкрива всички дългове на потребителя платени или не.

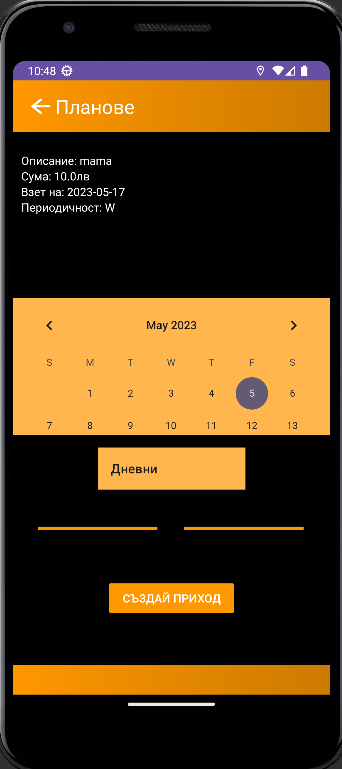


За създаването на нов дълг се попълват двете празни полета съответно с информация за дълга сумата за дълга и избора на бутона “Създай дълг”

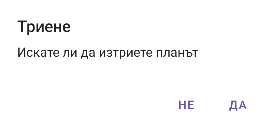
* + При избиране на стрелката в горния ляв ъгъл потребителя се отвежда до основното меню на приложението.
  + При избор на някой от дълговете се появява съобщение, което се обръща към потребителя за връщане на дълга.
* Спестявания екрана има за цел да покаже всички спестявани на потребителя - цел текущ баланс и др.:  
  
  + При избиране на стрелката в горния ляв ъгъл потребителя се отвежда до основното меню на приложението.

За създаване на дълг се изискват от потребителя отново описание на спестяването и цел (сума, която потребителя желае да постигне)

* + 
    - При натискане върху някое от спестяванията се появява съобщението по-горе, което позволява на потребителя да внесе или изтегли пари свързани с избраното спестяване.
      * При избора на внасяне се появява екрана посочен по-долу, който потребителя трябва да попълни със сумата желана за депозиране.
      * 
      * При избора на теглене и в случай че целта не е достигната се появява екрана посочен по-долу, който потребителят трябва да избере между “да” или “не”.
      * 
        + При избора на “да” се изтегля цялата сума събрана до момента.
        + При избора на “не” се прекратява операцията.
* Планове екрана има за цел да покаже всички планове на потребителя и да предостави начин за правене на нови



* + При избиране на стрелката в горния ляв ъгъл потребителя се отвежда до основното меню на приложението.
  + За създаване на дълг се изискват от потребителя отново да се въведе описание и сума но също и да се избере периодичност на планът и от коя дата да започне
    - при натискане на някой от плановете следното съобщение се появява



При натискане на бутона “ДА” планът ще бъде изтрит

При натискане на бутона ”Не” операцията по изтриване ще бъде прекратена

С това приключва ръководството за потребителя

# **Заключение**

Приложението се намира в стабилна версия която е незавършена и може да се използва само като демо версия или безплатно приложение с досегашните си възможности. Приложението е напълно достатъчно да удовлетвори желанията на нишата таргет група, а именно 14-23г. След допълнително развитие и добавяне на функционалности би могло да се превърне във водещо приложение за домашно счетоводство.

# **Използвана литература**

<https://infinitetrust.bg/8-te-nay-dobri-prilozheniya-za-lichni-finansi/?cn-reloaded=1>

<https://en.wikipedia.org/wiki/Java_(programming_language)>