Софтуерни архитектури и разработка на софтуер
Домашна работа № 3
Вариант 1 – ExpenseBuddy

Стефан Велев, № 62537
Даниел Халачев, № 62547

Допълнителни структури

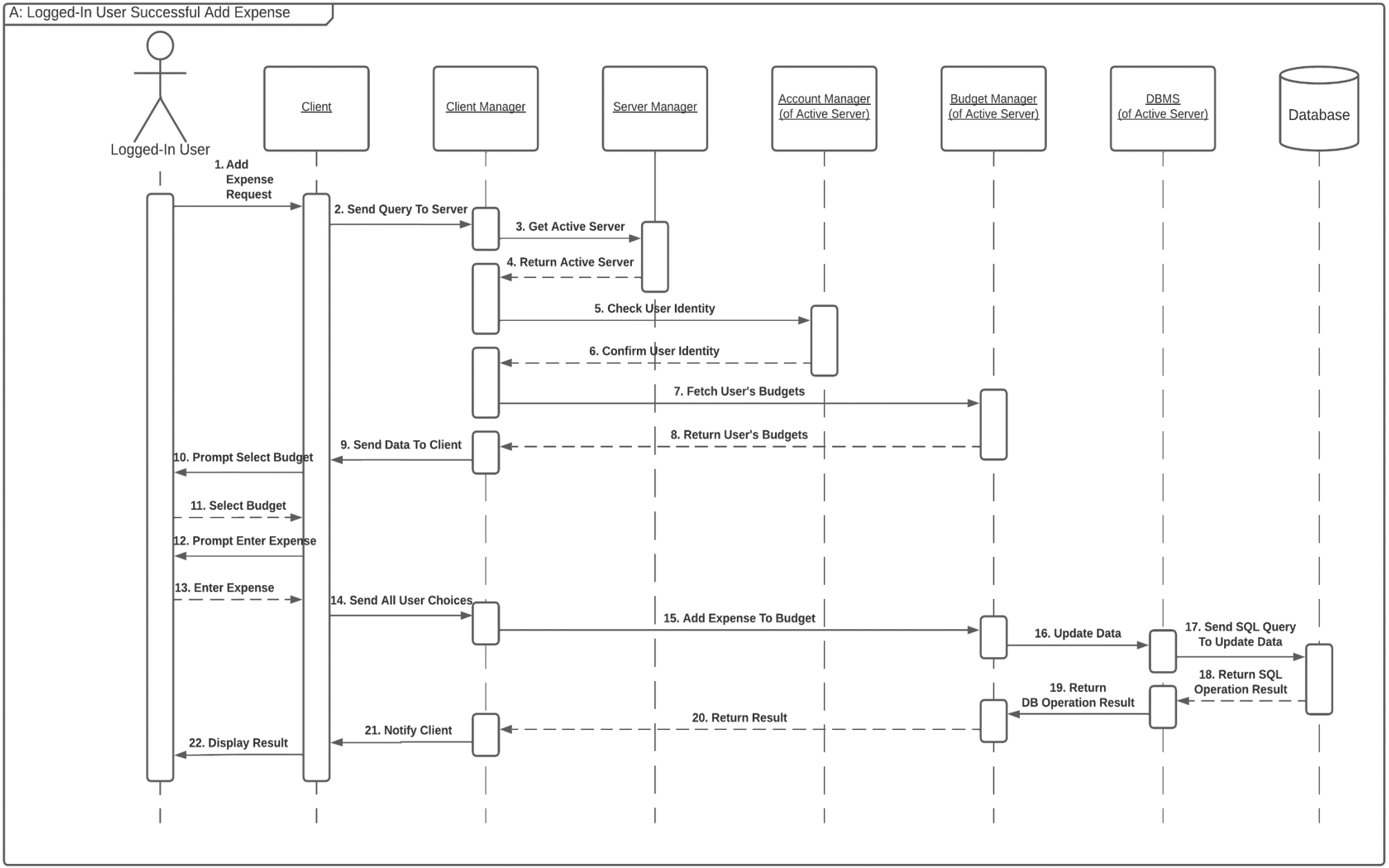
За проекта освен модулната декомпозиция, са от значителна важност също **Структурата на процесите** и **Структурата на разположението**.

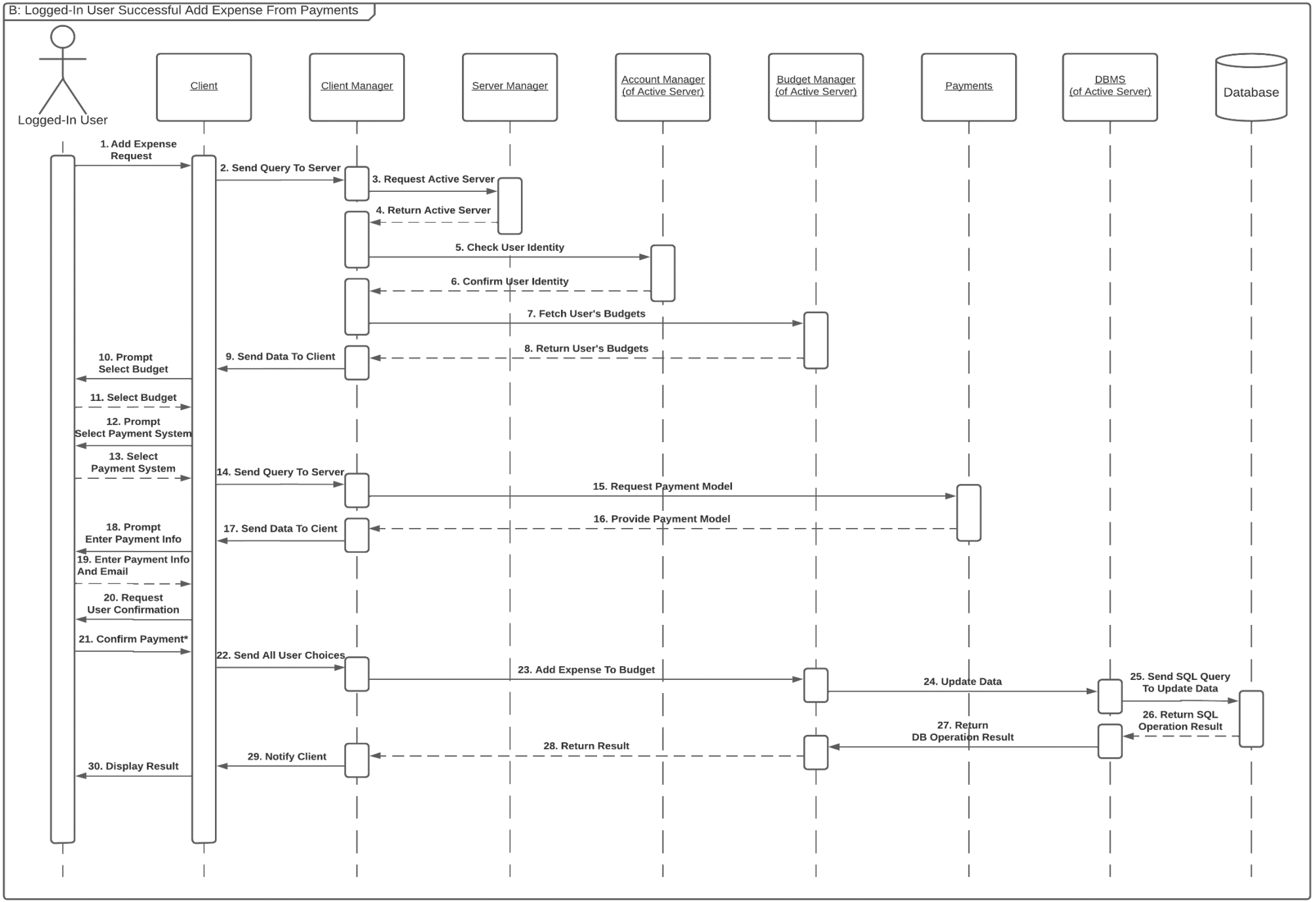
Структурата на процесите избрахме, защото поради множеството модули в системата настъпват множество взаимодействия между тях и операции, които за потребителя изглеждат атомарни (например добавяне на разход към бюджет), а всъщност са сложни и многоетапни.

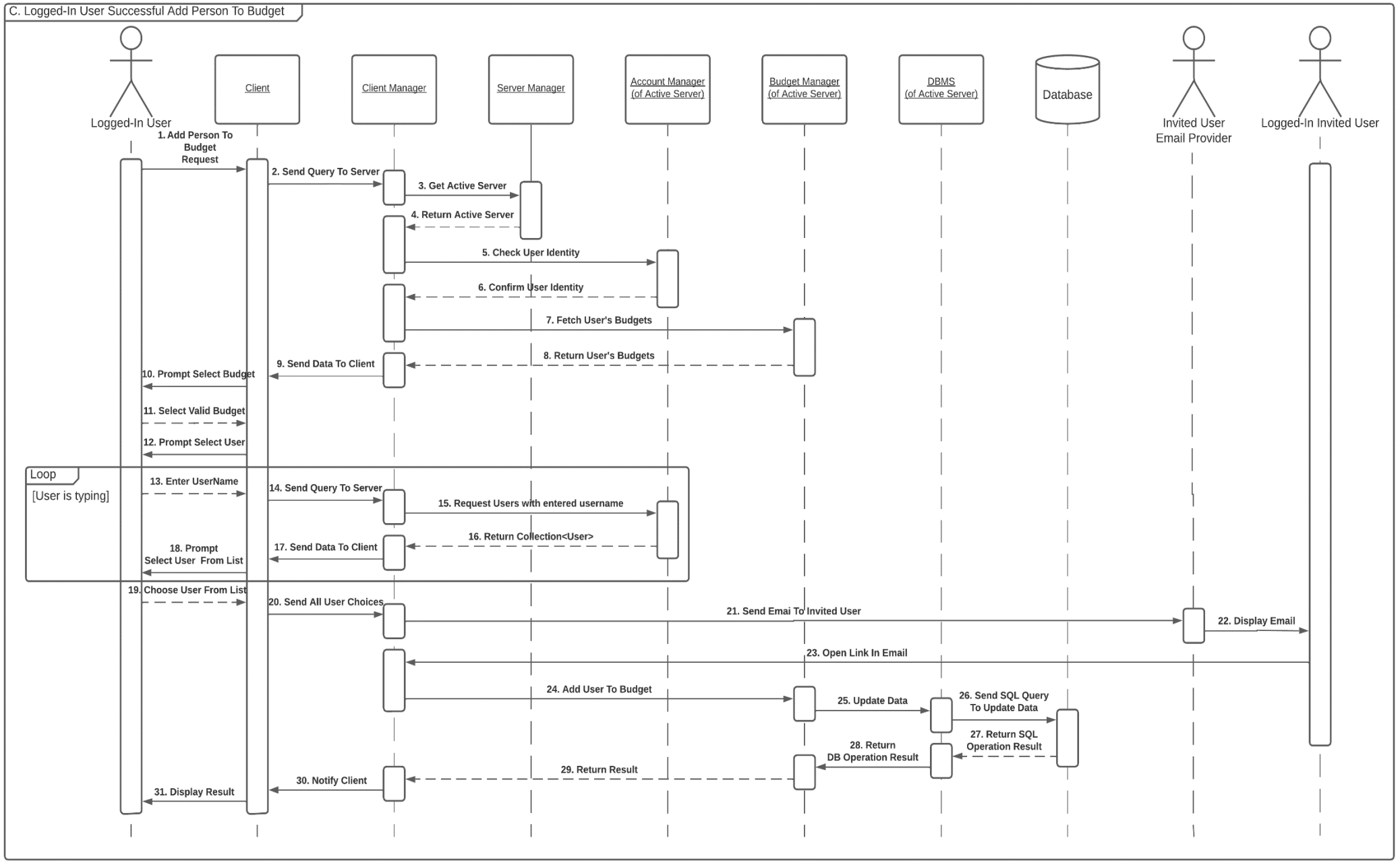
Предпочетохме **Структурата на разположението** пред Разпределение на работата поради избрания архитектурен стил – “Клиент – Сървър”, който предполага множество различни физически локации на хардуера и софтуера на системата. Без описанието на **Структурата на разположението**, би могло да настъпи неразбиране или двусмислие по отношение на това коя част от софтуера на какъв хардуер се намира.

1. Структура на процесите

a. Първично представяне


A: Logged-In User Successful Add Expense
Logged-In User
Client
Client Manager
Server Manager
Account Manager (of Active Server)
Budget Manager (of Active Server)
DBMS (of Active Server)
Database
1. Add Expense Request
2. Send Query To Server
3. Get Active Server
4. Return Active Server
5. Check User Identity
6. Confirm User Identity
7. Fetch User's Budgets
8. Return User's Budgets
9. Send Data To Client
10. Prompt Select Budget
11. Select Budget
12. Prompt Enter Expense
13. Enter Expense
14. Send All User Choices
15. Add Expense To Budget
16. Update Data
17. Send SQL Query To Update Data
18. Return SQL Operation Result
19. Return DB Operation Result
20. Return Result
21. Notify Client
22. Display Result


B: Logged-In User Successful Add Expense From Payments
Logged-In User
Client
Client Manager
Server Manager
Account Manager (of Active Server)
Budget Manager (of Active Server)
Payments
DBMS (of Active Server)
Database
1. Add Expense Request
2. Send Query To Server
3. Request Active Server
4. Return Active Server
5. Check User Identity
6. Confirm User Identity
7. Fetch User's Budgets
8. Return User's Budgets
9. Send Data To Client
10. Prompt Select Budget
11. Select Budget
12. Prompt Select Payment System
13. Select Payment System
14. Send Query To Server
15. Request Payment Model
16. Provide Payment Model
17. Send Data To Cient
18. Prompt Enter Payment Info
19. Enter Payment Info And Email
20. Request User Confirmation
21. Confirm Payment*
22. Send All User Choices
23. Add Expense To Budget
24. Update Data
25. Send SQL Query To Update Data
26. Return SQL Operation Result
27. Return DB Operation Result
28. Return Result
29. Notify Client
30. Display Result


C. Logged-In User Successful Add Person To Budget
Logged-In User
Client
Client Manager
Server Manager
Account Manager (of Active Server)
Budget Manager (of Active Server)
DBMS (of Active Server)
Database
Invited User Email Provider
Logged-In Invited User
1. Add Person To Budget Request
2. Send Query To Server
3. Get Active Server
4. Return Active Server
5. Check User Identity
6. Confirm User Identity
7. Fetch User's Budgets
8. Return User's Budgets
9. Send Data To Client
10. Prompt Select Budget
11. Select Valid Budget
12. Prompt Select User
Loop
[User is typing]
13. Enter UserName
14. Send Query To Server
15. Request Users with entered username
16. Return Collection<User>
17. Send Data To Client
18. Prompt Select User From List
19. Choose User From List
20. Send All User Choices
21. Send Emai To Invited User
22. Display Email
23. Open Link In Email
24. Add User To Budget
25. Update Data
26. Send SQL Query To Update Data
27. Return SQL Operation Result
28. Return DB Operation Result
29. Return Result
30. Notify Client
31. Display Result

b. Описание на елементите и връзките

№	Съобщение	Събитие	Обосновка
A1	Add Expense Request	Потребителят натиска елемент на графичния интерфейс, който започва операцията по добавяне на разход към бюджет. Модулът Client започва да изисква от сървъра необходимата информация	Модулът Client съдържа в имплементацията си инструкциите за добавяне на разход към бюджет (кои елементи от графичния интерфейс да покаже; директно изискване на бюджети). Така се избягва питането на сървъра за поредицата от действия, които трябва да се извършат, и се ускорява действието. Допитването кой е активният сървър става само веднъж, защото заявката се извършва за твърде кратко време, за да бъде ново допитване ефективно. Потвърждаването на самоличността става само веднъж, защото разговорът между Client и Server се осъществява като диалог чрез HTTP Request и HTTP Response и има начини за следене на потребителските сесии. След конкретизиране и на бюджета, и на разхода от страна на клиента, цялата информация се изпраща накуп, за да не се налага ClientManager да я съхранява до този момент. В противен случай ClientManager би бил претоварен модул, изпълняващ твърде много отговорности, но понеже той "посреща" трафика, това би довело до забавено
A2	Send Query To Server	Изпраща се заявка до сървъра чрез HTTP Request за предоставяне на необходимата информация. Модулът ClientManager свежда HTTP заявката до ООП класовете в системата	
A3	Get Active Server	ClientManager изисква от ServerManager информация кой е активният сървър	
A4	Return Active Server	ServerManager връща информация кой е активният сървър	
A5	Check User Identity	ClientManager проверява идентичността на потребителя преди да изисква данните	
A6	Confirm User Identity	AccountManager потвърждава (или отхвърля) идентичността на потребителя, изпратил заявката	
A7	Fetch User's Budgets	Сега ClientManager може да извика необходимите методи на BudgetManager, които връщат всички бюджети на този потребител	
A8	Return User's Budgets	BudgetManager връща структура, съдържаща всички бюджети на този потребител	
A9	Send Data to Client	ClientManager превръща тези инстанции на класове в JSON формат и връща отговор на клиента чрез HTTPResponse	
A10	Prompt Select Budget	Модулът Client представя информацията за бюджетите на потребителя и го подканва да избере един от тях	
A11	Select Budget	Потребителят избира един бюджет чрез графичния интерфейс	
A12	Prompt Enter Expense	Модулът Client предоставя графичен интерфейс за ръчно въвеждане на разход	
A13	Enter Expense	Потребителят въвежда информацията за един разход в предоставения графичен интерфейс	
A14	Send All User Choices	Модулът Client изпраща накуп цялата информация, необходима за операцията - конкретен бюджет и конкретен разход	
A15	Add Expense To Budget	ClientManager превръща информацията в обекти и извиква необходимите методи на BudgetManager с тези обекти като параметри	
A16	Update Data	BudgetManager добавя бюджета, но това трябва да бъде отразено в	

		базата данни. Изпраща се подкана да се обнови базата данни	изпълнение на заявките. Рискът за сигурността на данните е минимален, защото ще се прилага криптиране, а освен това съобщението ще съдържа цялостна информация само за разхода. Бюджетът ще бъде представен само чрез своето ID, което не носи информация за съдържанието.
A17	Send SQL Query To Update Data	Модулът DBMS приема новата информация и изпраща SQL заявка до базата данни	
A18	Return SQL Operation Result	Базата данни връща отговор дали заявката се е изпълнила успешно	
A19	Return DB Operation Result	Модулът DBMS връща информация дали операцията е минала успешно	
A20	Return Result	BudgetManager връща информация на ClientManager дали добавянето на бюджета е минало успешно	
A21	Notify Client	ClientManager превръща резултата в HTTP Response и го изпраща на клиента	
A22	Display Result	Модулът Client форматира пристигналия HTTP Response в четим за потребителя формат и го показва, за да го информира	

№	Съобщение	Събитие	Обосновка
B1	Add Expense Request	Потребителят натиска елемент на графичния интерфейс, който започва операцията по добавяне на разход към бюджет. Модулът Client започва да изисква от сървъра необходимата информация	Модулът Client съдържа в имплементацията си инструкциите за добавяне на разход към бюджет (кои елементи от графичния интерфейс да покаже; директно изискване на бюджети). Така се избягва питането на сървъра за поредицата от действия, които трябва да се извършат, и се ускорява действието. Допитването кой е активният сървър става само веднъж, защото заявката се извършва за твърде кратко време, за да бъде ново допитване ефективно. Потвърждаването на самоличността става само веднъж, защото разговорът между Client и Server се
B2	Send Query To Server	Изпраща се заявка до сървъра чрез HTTP Request за предоставяне на необходимата информация. Модулът ClientManager свежда HTTP заявката до ООП класовете в системата.	
B3	Get Active Server	ClientManager изисква от ServerManager информация кой е активният сървър	
B4	Return Active Server	ServerManager връща информация кой е активният сървър	
B5	Check User Identity	ClientManager проверява идентичността на потребителя преди да изисква данните	
B6	Confirm User Identity	AccountManager потвърждава (или отхвърля) идентичността на потребителя, изпратил заявката	
B7	Fetch User's Budgets	Сега ClientManager може да извика необходимите методи на BudgetManager, които връщат всички бюджети на този потребител	
B8	Return User's Budgets	BudgetManager връща структура, съдържаща всички бюджети на този потребител	
B9	Send Data to Client	ClientManager превръща тези инстанции на класове в JSON формат и връща отговор на клиента чрез HTTPResponse	

B10	Prompt Select Budget	Модулът Client представя информацията за бюджетите на потребителя и го подканва да избере един от тях	осъществява като диалог чрез HTTP Request и HTTP Response и има начини за следене на потребителските сесии. Всяка система за плащане използва различен формат за документиране на плащането. Затова е необходимо допитване до модула Payments какъв точно е този формат. Като резултат може да се върне списък с необходими данни или дори хипервръзка, която потребителят да отвори и да го препрати към уебсайт на системата за плащане. След конкретизиране и на бюджета, и на разхода от страна на клиента, цялата информация се изпраща накуп, за да не се налага ClientManager да я съхранява до този момент. В противен случай ClientManager би бил претоварен модул, изпълняващ твърде много отговорности, но понеже той “посреща” трафика, това би довело до забавено изпълнение на заявките. Рискът за сигурността на данните е минимален, защото ще се прилага криптиране, а освен това съобщението ще
B11	Select Budget	Потребителят избира един бюджет чрез графичния интерфейс	
B12	Prompt Select Payment System	Модулът Client подканва потребителя да избере система за плащане от предварително програмиран списък	
B13	Select Payment System	Потребителят избира система за плащане от списъка	
B14	Send Query To Server	Модулът Client изпраща заявка до сървъра за модел на плащането с избраната система за плащане (т.е какви данни се използват в тази система за плащането)	
B15	Request Payment Model	ClientManager превръща HTTP заявката в обекти и извиква методите на модула Payments с тях	
B16	Provide Payment Model	Модулът Payments връща информацията за модела на плащане	
B17	Send Data To Client	ClientManager превръща информацията в JSON формат и изпраща HTTP Response	
B18	Prompt Enter Payment Info	ClientManager подканва потребителя да въведе информацията за плащане по модела, който Payments е предоставил	
B19	Enter Payment Info And Email	Потребителят въвежда информацията. Валидацията става чрез графичния интерфейс в Client.	
B20	Request User Confirmation	Client изпраща молба за потвърждение до потребителя (чрез имейл или изскачащ прозорец)	
B21	Confirm Payment	Потребителят потвърждава плащането: 1. Ако е чрез имейл, чрез кликване върху линк в имейла, който препраща към графичния интерфейс на Client 2. Ако е чрез изскачащ прозорец – чрез натискане на елемент от графичния интерфейс	
B22	Send All User Choices	След потвърждаване на плащането, информацията за него се изпраща до сървъра.	
B23	Add Expense To Budget	ClientManager превръща информацията в обекти и извиква необходимите методи на BudgetManager с тези обекти като параметри	
B24	Update Data	BudgetManager добавя бюджета, но това трябва да бъде отразено в Базата данни. Изпраща се подкана да се обнови базата данни	
B25	Send SQL Query To Update Data	Модулът DBMS приема новата информация и изпраща SQL заявка до базата данни	

B26	Return SQL Operation Result	Базата данни връща отговор дали заявката се е изпълнила успешно	съдържа цялостна информация само за разхода. Бюджетът ще бъде представен само чрез своето ID, което не носи информация за съдържанието.
B27	Return DB Operation Result	Модулът DBMS връща информация дали операцията е минала успешно	
B28	Return Result	BudgetManager връща информация на ClientManager дали добавянето на бюджета е минало успешно	
B29	Notify Client	ClientManager превръща резултата в HTTP Response и го изпраща на клиента	
B30	Display Result	Модулът Client форматира пристигналия HTTP Response в четим за потребителя формат и го показва, за да го информира	

№	Съобщение	Събитие	Обосновка
C1	Add Expense Request	Потребителят натиска елемент на графичния интерфейс, който започва операцията по добавяне на разход към бюджет. Модулът Client започва да изисква от сървъра необходимата информация	Модулът Client съдържа в имплементацията си инструкциите за добавяне на потребител към бюджет (кои елементи от графичния интерфейс да покаже; директно изискване на бюджетите на потребителя, допитване за потребители на базата на критерии за търсене). Така се избягва питането на сървъра за поредицата от действия, които трябва да се извършат, и се ускорява действието. Допитването кой е активният сървър става само веднъж, защото заявката се извършва за твърде кратко време, за да бъде ново допитване ефективно. Потвърждаването на самоличността става само веднъж, защото разговорът между Client и
C2	Send Query To Server	Изпраща се заявка до сървъра чрез HTTP Request за предоставяне на необходимата информация. Модулът ClientManager свежда HTTP заявката до ООП класовете в системата.	
C3	Get Active Server	ClientManager изисква от ServerManager информация кой е активният сървър	
C4	Return Active Server	ServerManager връща информация кой е активният сървър	
C5	Check User Identity	ClientManager проверява идентичността на потребителя преди да изисква данните	
C6	Confirm User Identity	AccountManager потвърждава (или отхвърля) идентичността на потребителя, изпратил заявката	
C7	Fetch User's Budgets	Сега ClientManager може да извика необходимите методи на BudgetManager, които връщат всички бюджети на този потребител	
C8	Return User's Budgets	BudgetManager връща структура, съдържаща всички бюджети на този потребител	
C9	Send Data to Client	ClientManager превръща тези инстанции на класове в JSON формат и връща отговор на клиента чрез HTTPResponse	
C10	Prompt Select Budget	Модулът Client представя информацията за бюджетите на потребителя и го подканва да избере един от тях (като предварително е филтрирал тези, които са достигнали лимита за споделяне от 5ма души)	
C11	Select Valid Budget	Потребителят избира един бюджет чрез графичния интерфейс	

C12	Prompt Select User	Client подканва потребителя да въведе потребител	Server се осъществява като диалог чрез HTTP Request и HTTP Response и има начини за следене на потребителските сесии. След конкретизиране и на бюджета, и на потребителя от страна на клиента, цялата информация се изпраща накуп, за да не се налага ClientManager да я съхранява до този момент. В противен случай ClientManager би бил претоварен модул, изпълняващ твърде много отговорности, но понеже той "посреща" трафика, това би довело до забавено изпълнение на заявките. Рискът за сигурността на данните е минимален, защото ще се прилага криптиране, а освен това съобщението ще съдържа цялостна информация само за разхода. Бюджетът ще бъде представен само чрез своето ID, което не носи информация за съдържанието.
C13	Enter UserName	Потребителят започва да въвежда информация за човека, с когото иска да сподели бюджета	
C14	Send Query To Server	Client изпраща заявка до сървъра за списък от всички потребители, които отговарят на въведените от потребителя данни	
C15	Request Users with Entered Usernames	ClientManager преработва заявката до инстанции на обекти и извиква необходимите методи на AccountManager, които връщат списък от потребители, отговарящи на критериите	
C16	Return Collection<User>	AccountManager връща списък на потребителите, които отговарят на критериите за търсене	
C17	Send Data To Client	ClientManager форматира до JSON формат данните и ги изпраща на клиента	
C18	Prompt Select User From List	Client разчита данните и ги представя на потребителя за избор. Потребителят може да избере човек от предоставения списък или да продължи да пише информация в полето/полетата за търсене докато не получи удовлетворителен резултат. Ако потребителят продължи да пише, действията C13-C18 се повтарят циклично.	
C19	Choose User From List	Потребителят избира човек от предоставения списък.	
C20	Send All User Choices	Модулът Client изпраща накуп цялата информация, необходима за операцията - конкретен бюджет и конкретен потребител за добавяне	
C21	Send Email To Invited User	Client Manager изпраща имейл до поканения потребител (до неговия Email Provider)	
C22	Display Email	Email Provider (част от обкръжението) представя полученото съобщение (имейл), използвайки собствена имплементация и интерфейс (външни за системата), на поканения потребител	
C23	Open Link In Email	Поканеният потребител отваря хипервръзката, която пренасочва към сървъра и извършва заявката, добавяща го към бюджета	
C24	Add User To Budget	ClientManager извършва заявката, като превръща информацията в обекти и извиква необходимите методи на BudgetManager с тези обекти като параметри	
C25	Update Data	BudgetManager добавя потребителя към бюджета, но това трябва да бъде отразено в Базата данни. Изпраща се подкана да се обнови базата данни	

C26	Send SQL Query To Update Data	Модулът DBMS приема новата информация и изпраща SQL заявка до базата данни	
C27	Return SQL Operation Result	Базата данни връща отговор дали заявката се е изпълнила успешно	
C28	Return DB Operation Result	Модулът DBMS връща информация дали операцията е минала успешно	
C29	Return Result	BudgetManager връща информация на ClientManager дали добавянето на потребител към бюджета е минало успешно	
C30	Notify Client	ClientManager превръща резултата в HTTP Response и го изпраща на клиента	
C31	Display Result	Модулът Client форматира пристигналия HTTP Response в четим за потребителя формат и го показва, за да го информира	

c. Описание на обкръжението

Добавянето на потребител към бюджет изисква потвърждение от втория потребител, което се осъществява чрез имейл. Следователно потвърждението зависи от външна система - доставчик на електронна поща.

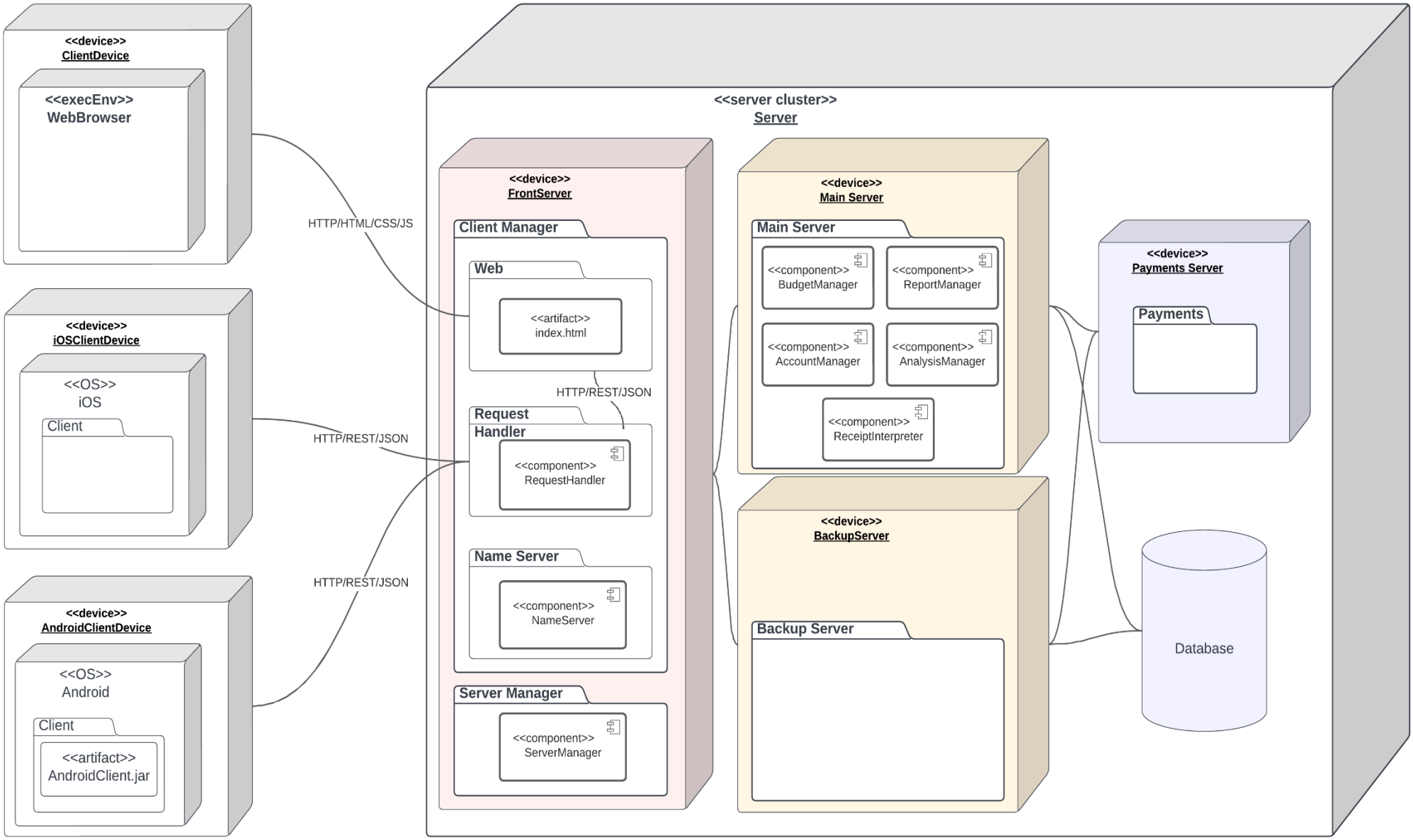
Добавянето на разход към бюджет чрез външни системи зависи от тези външни системи. При неизправност вътре в тях, настъпва неизправност и в тази система. За щастие неизправността тук ще е само временната липса на тази функционалност, а има и други две възможности за добавяне на разход към бюджет.

d. Описание на възможните вариации

Разположението на различните процеси върху различен хардуер се влияе непосредствено от разположението на модулите, в които се извършват тези процеси, върху хардуера. Невъзможно е процес, предизвикан от даден модул, да се намира на хардуер, различен от този на модула. Разпределението на модулите върху хардуера и възможните вариации са описани в **3.2. Структура на разположението.**

2. Структура на разположението

a. Първично представяне



Диаграма: UML диаграма на внедряването на цялата система

b. Описание на елементите и връзките

Системата ще бъде изградена основно в стила Клиент-Сървър. Следователно тя ще бъде разположена на две основни физически места:

- Машината на потребителя (клиента)
    - Вариант iOS

    IOS клиентът ще представлява приложение, което е написано на Swift и ще се изпълнява на операционната система iOS. Това мобилно приложение ще съдържа инстанция на модула Client в себе си.

    - Вариант Android

    Android клиентът, подобно на iOS клиентът, ще съдържа приложение, написано да се изпълнява на операционната система iOS и ще съдържа екземпляр на модула Client в себе си. Машината ще съдържа като артефакт програмни файлове на Java/Kotlin

    - Вариант Web Browser

    Машината на клиента няма как да съдържа код за клиентската част. За целта, достъпът ще се осъществява чрез уеб браузър, кодът за който ще се помещава на сървъра в модула Web, който е аналогичен на модула Client в първите два варианта.

И трите модула ще изпращат HTTP заявки до сървъра (в REST стил и данни в JSON формат), по-точно до модула ClientManager.RequestHandler в сървърната част, който ще ги обработва.

- Сървърът

Сървърната част ще представлява клъстър от няколко сървъра (физически устройства), чиято взаимна зависимост е сведена до минимум:

- Front Server - посреща потребителския трафик и управлява сървърите за бизнес логиката

  Съдържа в себе си модулите:
    - ClientManager – осигурява Web клиентът на системата; посреща потребителските заявки и ги свежда до ООП модела на MainServer и BackupServer
    - ServerManager - контролира състоянието на сървърите за бизнес логика и осигурява надеждността на системата
- Main Server - изпълнява бизнес логиката и съдържа модулът MainServer
- BackupServer - изпълнява бизнес логиката, когато BackupServer е извън строя; съдържа модулът BackupServer
- PaymentsServer - Помещава единствено модулът Payments, за да се осигури максимална надеждност на работата с външни системи за плащания и за да може да се ползва и от MainServer, и от BackupServer и да не се налага дублирането му
- Database - помещава базата данни

c. Описание на обкръжението

Архитектурата “клиент-сървър” е надеждна и ефективна, но и тя има недостатък - за функционирането ѝ е необходима надеждна интернет връзка. Този недостатък не може да бъде преодолян поради същността на системата - да обслужва множество клиенти от разстояние. Единственото, което системата може да гарантира, е предвиждане и обработка на грешките във всички случаи, когато се прекъсне връзката между клиент и сървър, за да се осъществи бързо възстановяване по-късно. Аналогична е ситуацията между сървъра и системите за външни плащания. Тяхната работоспособност не зависи от тази система и в случай на отказ на някоя тях, системата ExpenseBuddy ще е принудена да изчака до отстраняването на неизправността. За щастие, в този случай всички останали функционалности освен плащането чрез външна система ще бъдат налични.

d. Описание на възможните вариации

- Модулът BackupServer дублира функционалността на MainServer, но не и имплементацията му. Следователно физическото устройство Backup Server може да предоставя по-малко изчислителни мощности от Main Server.
- С цел максимално разделение на функционалността е възможно модулът Web да се помещава на свое собствено физическо устройство (сървър), но това решение е оправдано само ако:
    - потребителите използват предимно Web клиенти за сметка на мобилни приложения
    - Backend--ът на Web клиента изисква значителни изчислителни мощности
    - Има финансови възможности за осигуряването на ново устройство (сървърите са скъпа техника)
- Възможно е модулът Payments да бъде дублиран в Main Server и BackupServer и физическото устройство Payments Server да бъде премахнато (например поради финансови причини).
- Възможно е модулът Server Manager да бъде поместен в собствено физическо устройство, ако се окаже, че диагностиката на сървърите отнема значителни ресурси, от които ClientManager се нуждае повече.