ДОКУМЕНТАЦИЯ

**ТЕМА**: Проект „Система за търсене на коли“

***Изготвен от****: Калоян Димитров, Славян Христов и Радослав Иванов,*

*учащи в „Технически университет“ – Варна*

Информация за студентите:

*Студент 1: Студент 2:*

*име: Славян Христов име: Калоян Димитров*

*фак. номер: 18621793 фак.номер: 18621798*

*факултет: ФИТА факултет: ФИТА*

*специалност: СИТ специалност: СИТ*

*група: 5б група: 5б*

*Студент 3:*

*име: Радослав Иванов*

*фак. номер: 18621797*

*факултет: ФИТА*

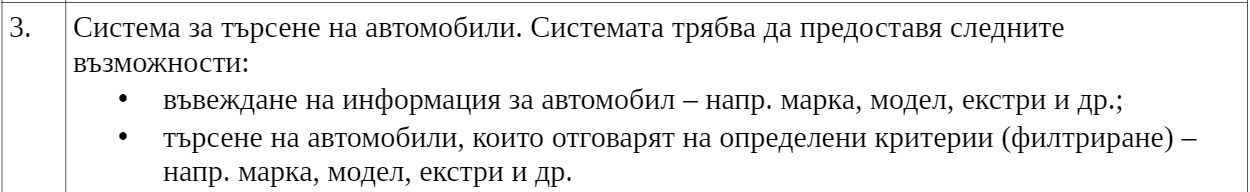
*специалност: СИТ*

*група: 5б*

Съдържание*:*

1. *Заглавна страница*
2. *Задание*
3. *Анализ на проблема*
4. *Използвани технологии*
5. *Описание на стъпките по реализацията*
6. *Обяснение на хода на програмата*
7. *Структура на програмата*
8. *Проектиране на системата*
9. *Изводи*

Задание



Анализ на проблема

1. Функционални изисквания

Функционалността на системата трябва да бъде съобразена с нуждите на потребителите, които ще я използват. Интерфейсът на системата трябва да бъде прост, приятен и да позволява лесно и безпрепятсвено използване, без грешки в процеса на работа.

Лицата, които ще използват систамата, са потребител и админстратор. Те трябва да могат да изпълняват следните действия:

1. Администратор

* влизане в системата
* добавяне на кола
* търсене на коли
* изтриване на кола
* преглед на кола

1. Потребител

* търсене на коли
* преглед на кола

Системата трябва да предостави възможност за изпъленение на всички действия описани горе. Допълнителни функционалности трябва да бъдат имплементирани за валидиране на данните при логване в системата и при добавяне на кола в системата.

Използвани технологии

1. JavaFX – Реализация на потребителски и графичен интерфейс
2. JPA + Hibernate – Реализация на слоя за работа с база данни
3. MySQL database – база данни
4. Java – използван език
5. Log4J2 – за регистриране на събитията
6. JUnit Jupiter – за тестови резултати
7. Обосновка на избора за система за контрол на версиите

А) Причина на избора

[Git](https://en.wikipedia.org/wiki/Git) е система за контрол и история на версиите на кода, която се използва предимно от уеб разработчици и програмисти за разработката на приложения. Включително самият [проект на Git използва тази система](https://git.kernel.org/pub/scm/git/git.git/), като много разработчици извършват промени в кода му (C, Shell).Чрез Git е възможно един или много разработчици да прилагат промени в кода на файловете на един проект, по начин, чрез който се избягват несъответствия и конфликти от промените.

Б) Трудности при използването

Изпитвахме трудности при “merge” на бранчове, когато имаше кофликти, но успявахме да постигнем резултата. Друг проблем беше изпозлването на “pull”. От време на време се получаваха грешки и трябваше да изтриваме базата данни предварително преди използването на тази команда.

В) Настройки, нужни за работа с него

- инсталация и настройка за работа с “bash” и командите да се базират на операционна система Линукс

Г) Съпоставка на избрания софтуер с други

В момента едни от най-известните централизирани version control системи са Subversion, TFVC, Perforce and Clearcase. Използвайки тези системи, разработчиците достъпват едно централно repository. Ако бъдат направени промени, те ще достигнат до всеки разработчик преди той/ тя да може да commit-не собствените си промени и, за съжаление, това включва и неработещ код. При наличието на единствено “истинско” repository, работата офлайн може да бъде предизвикателство. За да могат да бъдат изпълнени базови операции, като преглед на историята или commit-ване на кода, е необходим достъп до това repository.

Една от първите разпределени VCS е Bitkeeper (1997). В момента най-популярните такива системи са Git, GNU Arch и Mercurial. В разпределения модел всеки разработчик има собствено копие на repository-то. Използвайки този модел, разработчиците могат да работят офлайн – могат да commit-ват, създават нови или обединяват branch-ове, да виждат история и всичко друго, от което се нуждаят, след като имат имат цялото repository под ръка. Достъп до интернет е нужен само когато е необходима синхронизация с другите членове на [екипа](https://mentormate.com/blog/communication-managing-remote-teams/).

1. Обосновка на избора за система за управление на проекти

А) Причина на избора

* **JIRA** е уеб-базирана система за проследяване на грешки (bug tracking), проблеми (issue tracking) и управление на разработката на софтуерни проекти от [Atlassian Software Systems](https://bg.wikipedia.org/w/index.php?title=Atlassian_Software_Systems&action=edit&redlink=1). Управлението на технологичния процес (workflow) прави JIRA подходяща за управление и подобряване на процеси.
* JIRA притежава прекрасен и удобен интерфейс

Б) Трудности при използването

Изпитвахме трудности при отварянето на стари, завършени спринтове. Търсихме опция за това, но не намерихме. Единственият начин беше чрез „Work page”.

В) Настройки, нужни за работата с него

* направа на акаунт
* направа на проект и избор на методология
* изготвяне на спринтове

Г) Използвани функционалности

- изполване на „backlog” и спринтове за направа и разпределение на задачите

1. Обосновка и разяснения към избраната методология за разработка на софтуера

А) Причина на избора

Използвахме “Scrum”, защото е перфектен за работа в екип. Scrum процесът се състои от отделни итерации, наречени *спринтове*. Спринтовете могат да имат продължителност от една седмица до четири седмици. В края на всеки спринт екипът разполага с работеща версия на продукта, която включва всички готови задачи от backlog-а.

Б) Предимства на подхода

Предимството на подхода е, че при „Скръм“ , за разлика от „Waterfall”, за всяка една част от проекта се извършва планиране, проектиране, разработване, тестване и ревю. За всяка една част се изготвя отделен спринт и се добавят задачи, които се разпределят по участниците в отбора от „скръм мастъра“. По време на всеки спринт се организират събрания, в които се дискутират нуждите на клиента. След всеки спринт се организира събрание, в което се показва цялата работа, свършена до момента, на собственика на продукта.

В) Недостатъци на подхода, открити при изпълнението на задачата

Недостатък на този подход е, че не може да се определи точна крайна дата на проекта.

Г) Съпоставка на подхода с други

За разлика от „Waterfall”, за всяка една част от проекта се извършва планиране, проектиране, разработване, тестване и ревю. „Скръм“ методологията е по-гъвкава, която позволява лесно връщане към други, вече завършени, части на проекта за подобряване.

Спрямо „екстремно програмиране“, при „Скръм“ всеки работи индивидуално по задачата си, а при „екстремното програмиране“ се работи по двойки.

Описание на стъпките по реализацията

1. Определяне на роли

А) Славян Христов – front-end разроботчик, тестър, мениджър

Б) Калоян Димитров – back-end разроботчик, тестър, мениджър

В) Радослав Иванов – архитект, разроботчик, тестър

1. Разделяне на задачи

А) Славян Христов – стилизиране, работа с JavaFX, Junit Jupiter

Б) Калоян Димитров – работа с Hibernate + JPA, Junit Jupiter, работа с база данни

B) Радослав Иванов – работа с UML, помощ с разработката, изготвяне на документация и презентация

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер на задачата | Опсание | Предшестваща задача | Време | Изпълнител |
| 1 | **Анализ на изискванията** |  |  |  |
| 2 | Функционални |  | 2 дена | всички |
| 3 | Нефункционални |  | 1 ден | всички |
| 4 | **Проектиране** |  |  |  |
| 5 | Определяне на дизайна |  | 1 ден | Славян и Калоян |
| 6 | UML диаграми |  | 3 дена | Радослав |
| 7 | EER model |  | 1 ден | Славян И Калоян |
| 8 | **Разработка** |  |  |  |
| 9 | Front-end |  | 15 дена | Славян, Радослав |
| 10 | Back-end |  | 15 дена | Калоян, Радослав |
| 11 | Бизнес логика |  | 1 ден | Калоян |
| 12 | Тестване |  | 3 дена | Калоян, Славян, Радослав |
| 13 | **Документиране** |  |  |  |
| 14 | Документация |  | 2 дена | Радослав |
| 15 | Презентация |  | 1 ден | Радослав |

1. Определяне на срокове

Начална дата: 14/04/2021

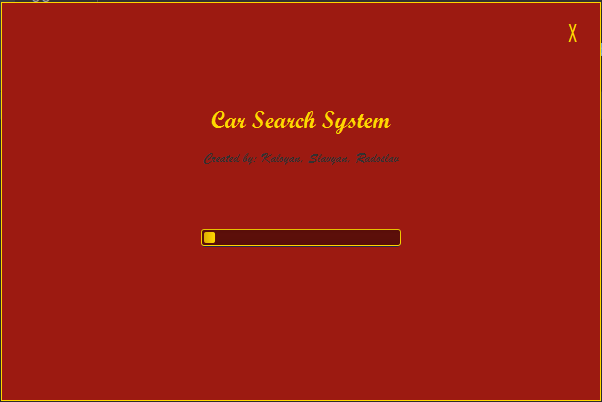
Анализ на изискванията: 16/04/2021

Проектиране: 21/04/2021

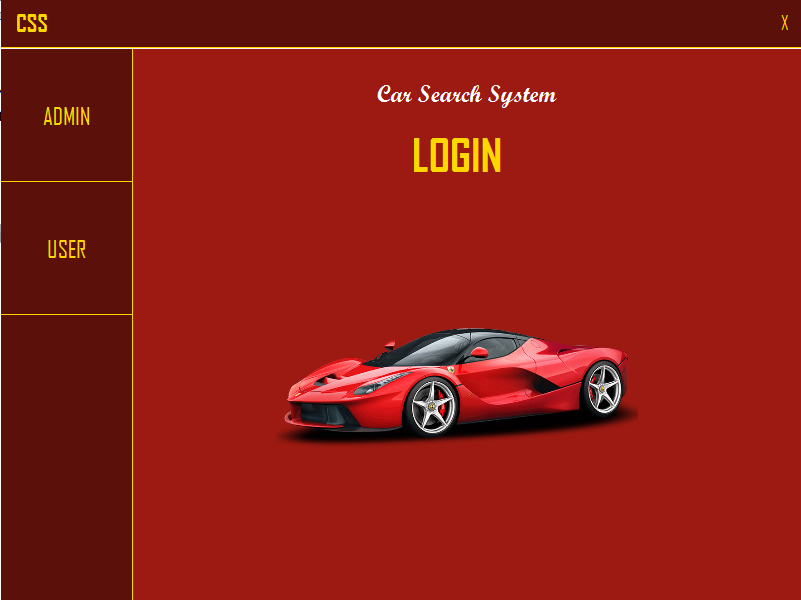
Разработка: 10/05/2021

Документиране: 13/05/2020

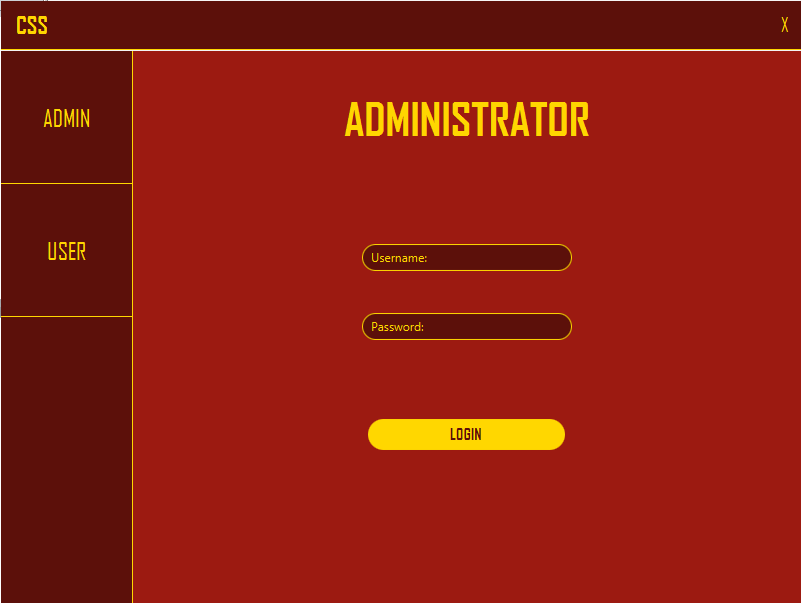
Обяснение на хода на програмата



При стартиране на програмата се появява прозорец за зареждане. Докато програмата зарежда, базата данни се изготвя и инициализира.



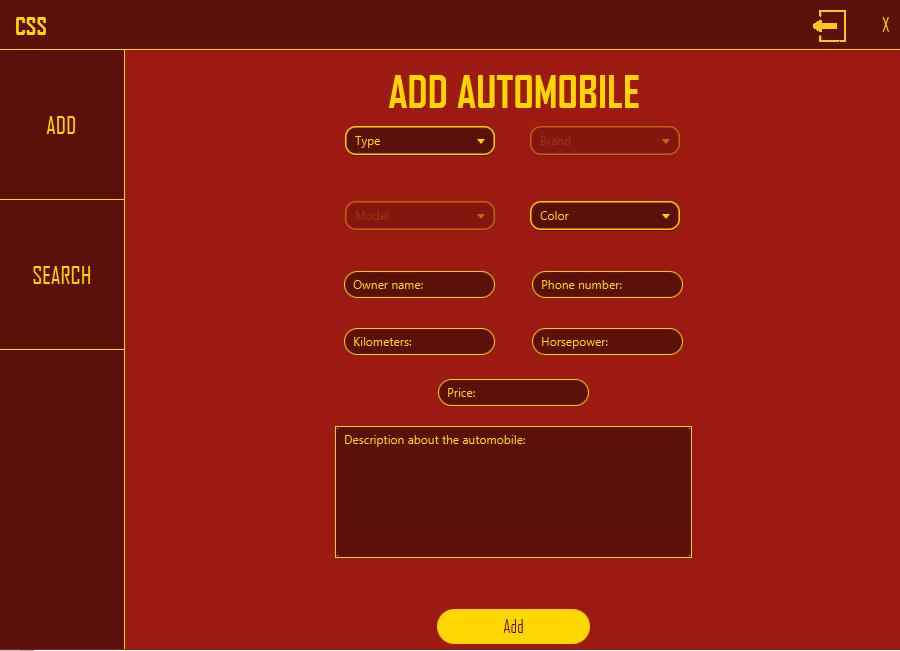
Когато зареди началният екран, потребителят може да избере като какъв да влезе в системата. Опциите, които може да избере са: Администратор и Потребител.



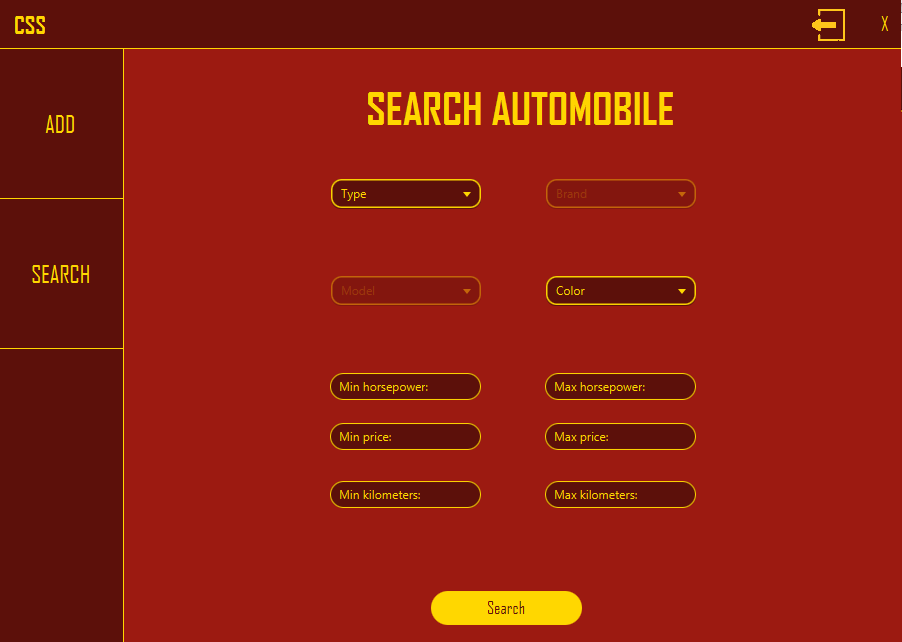
При влизане като „Администратор“ излиза прозорец за логване в системата.



Администраторът има право да вкарва коли вътре в системата и да ги търси.



Първо се въвеждат данните на колата. След натискане на бутона „добави“ се извършва валидация на данните. Ако данните са невалидни или са празни, излиза съобщение за грешка. Ако данните са валидни и попълнени, колата се добавя в базата данни.



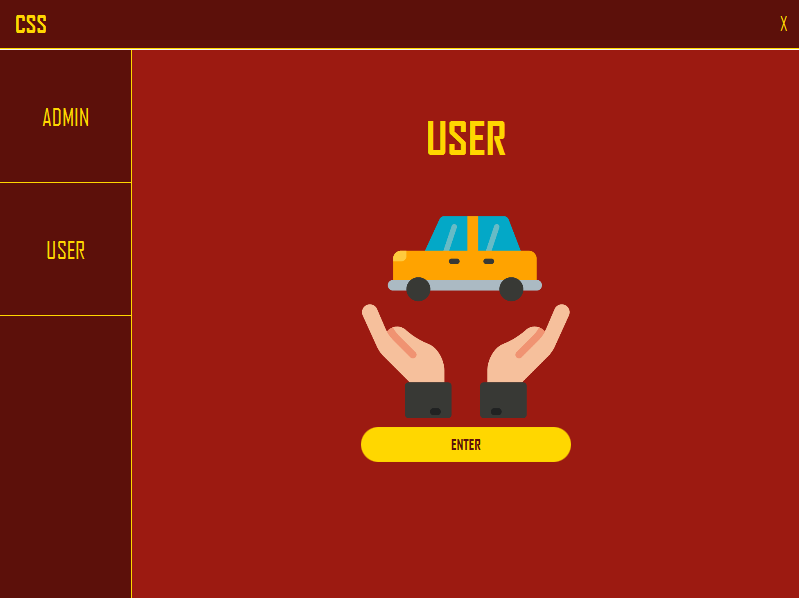
При търсене на автомобили се появява търсачка. Въвеждат се данни, по които потребителят иска да търси коли. След натискане на бутона „търси“ се появява нов прозорец с намерените коли.



След показването на автомобилите администраторът има право да погледне допълнителна информация за избрания автомобил или да го изтрие.

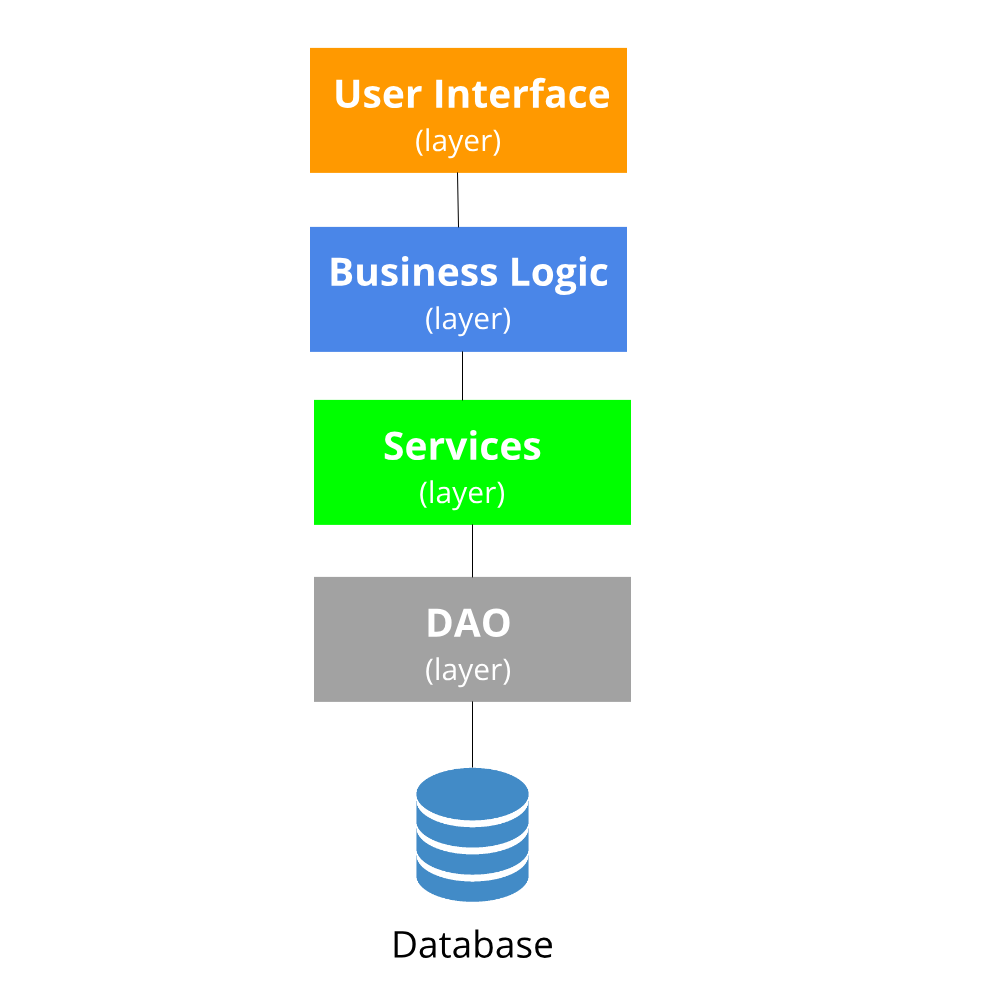
Търсачката е същата и за потребителя, но няма опция за изтриване.







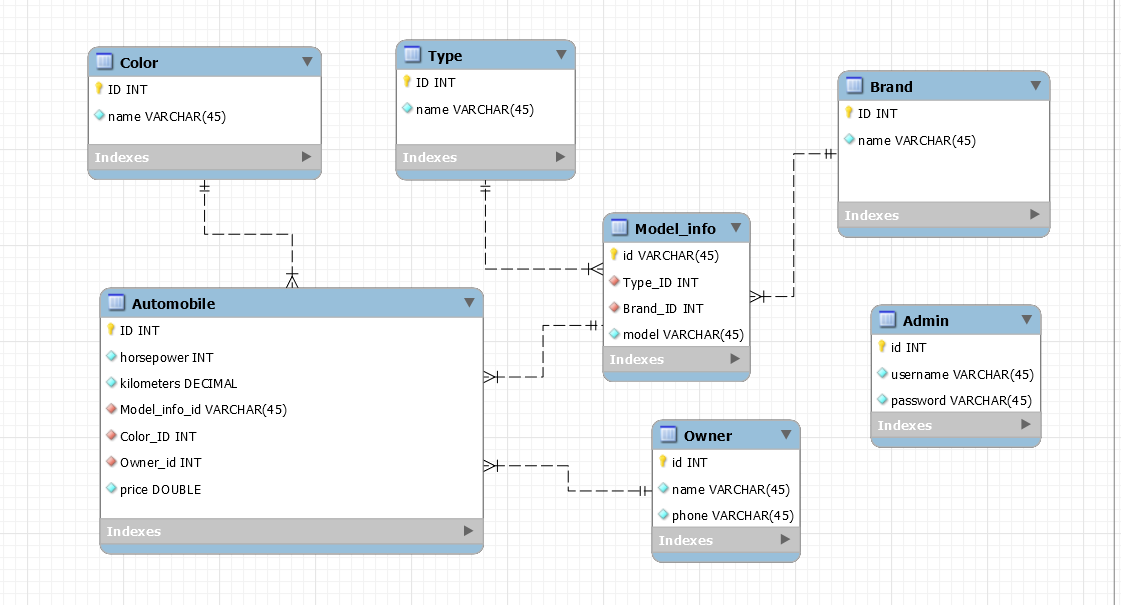
Структура на програмата

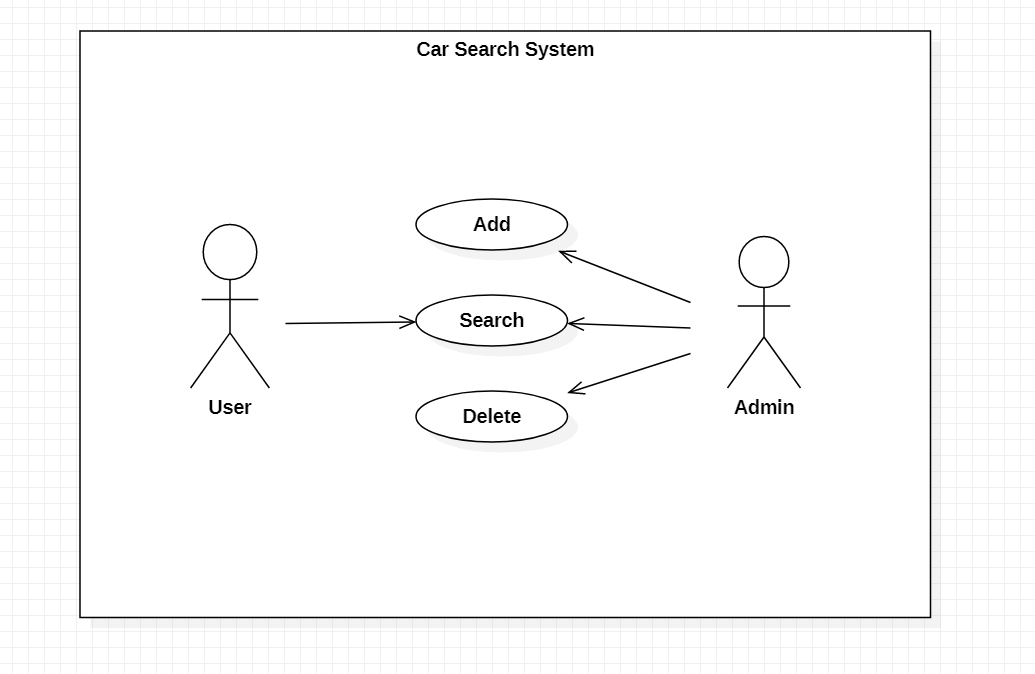


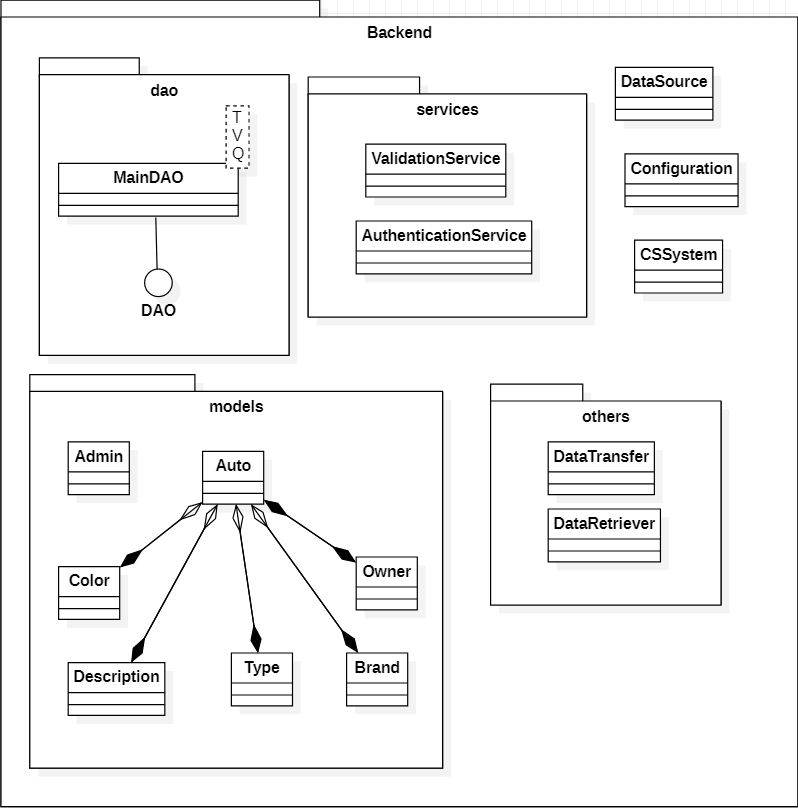
1. Дефиниция на модулите на системата
2. User Interface – Потребителски интерфейс, който служи за навигиране на потребителя в системата
3. Business Logic – Това е модулът, който служи за имплементация на бизнес идеите и правилата. Служи си с обекти от високо ниво и бизнес обекти и отразява ясна представа за работния процес на програмата.
4. Services – Това е модул от по-ниско ниво, който служи за извършване на нещо странично, което е нужно в процеса на работа
5. DAO(Data Access Object) – Тове е модул от по-ниско ниво, който служи само и единствено за извършване на операции с базата данни

Проектиране на системата

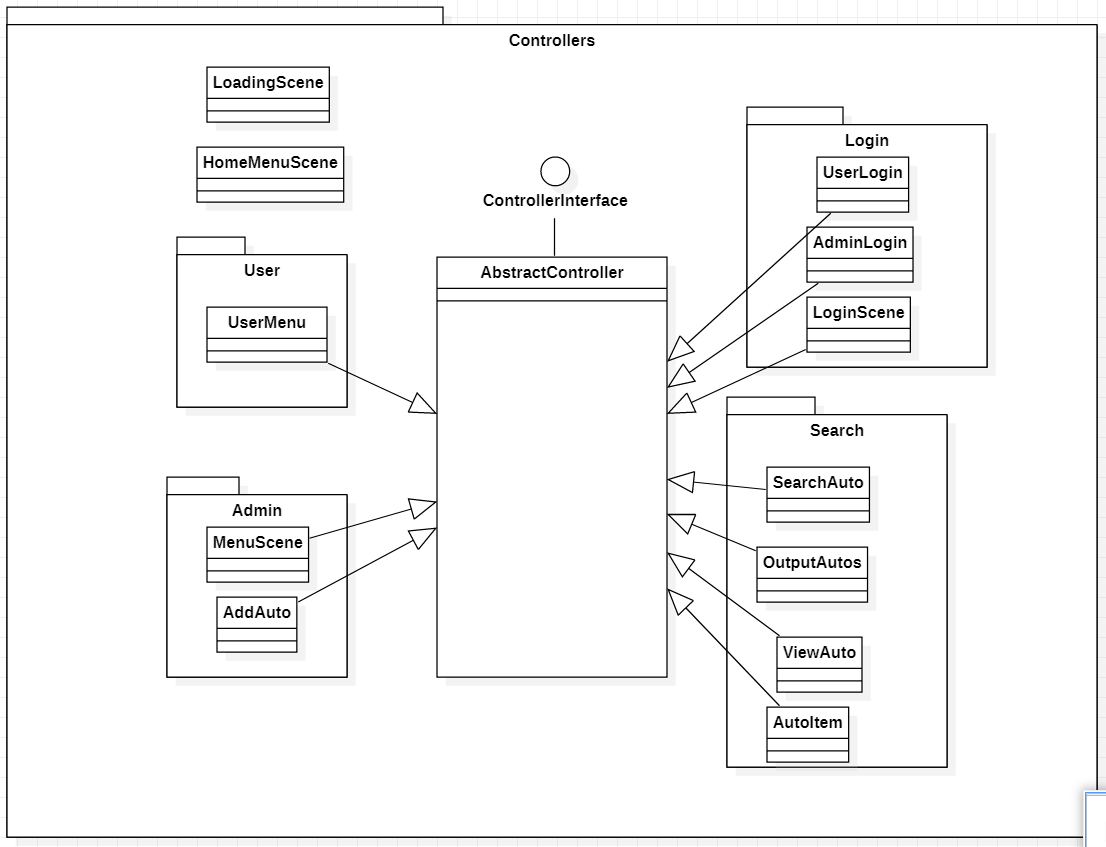
1. EER Model



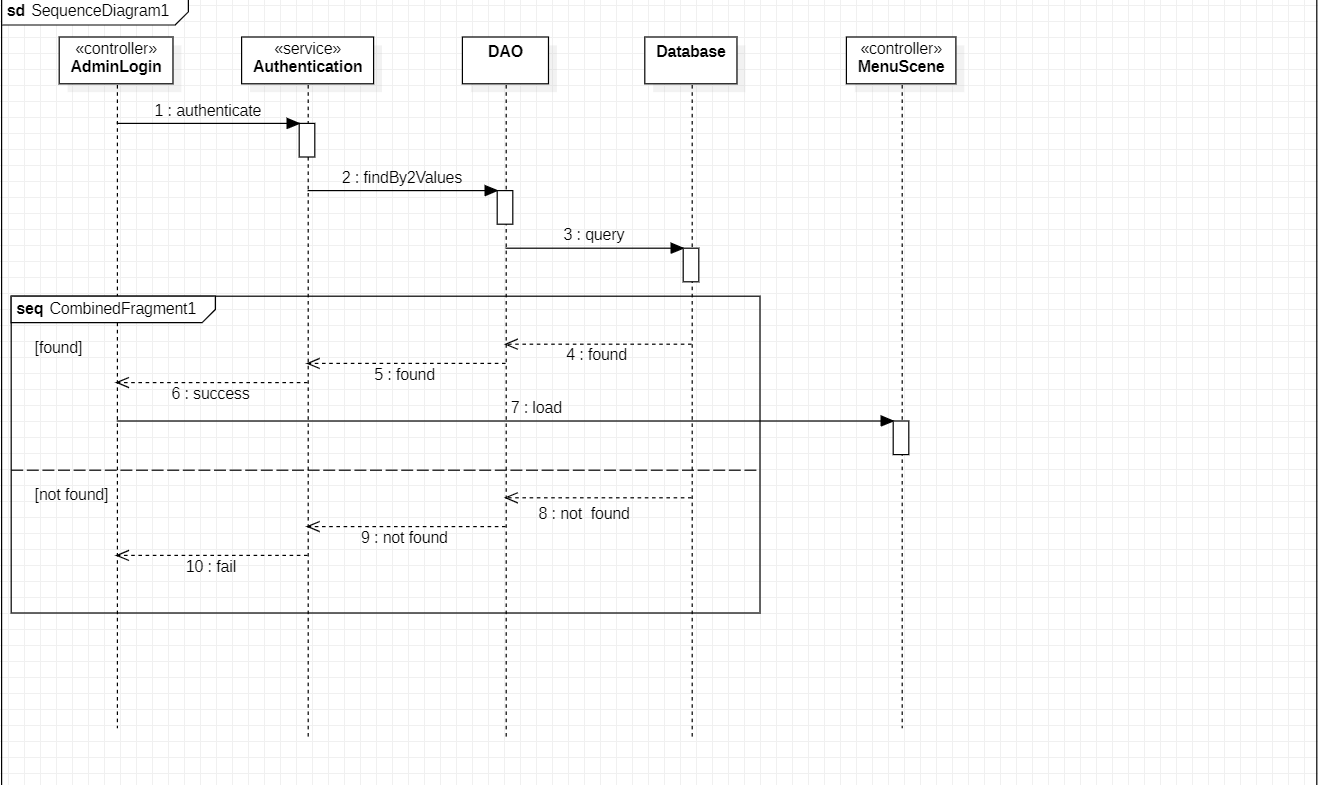
1. Use Case
2. Backend Class Diagram



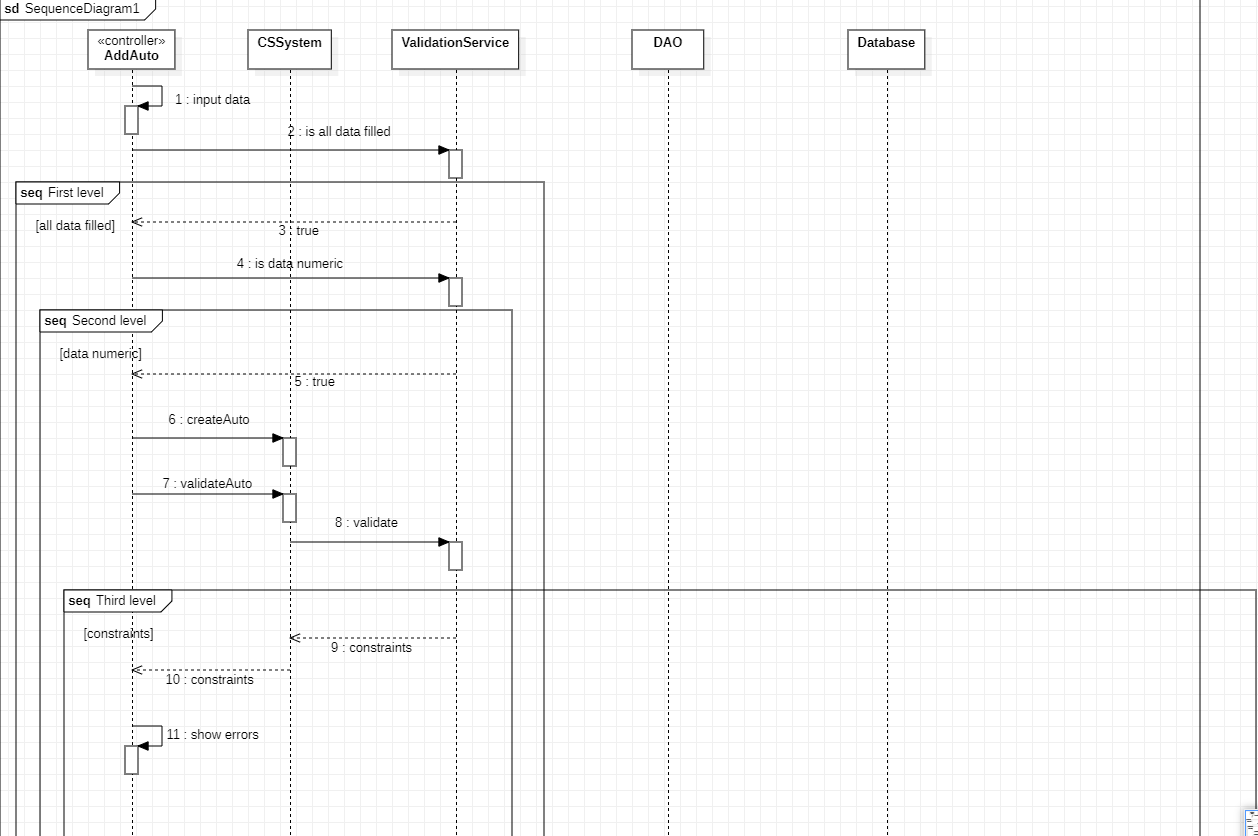
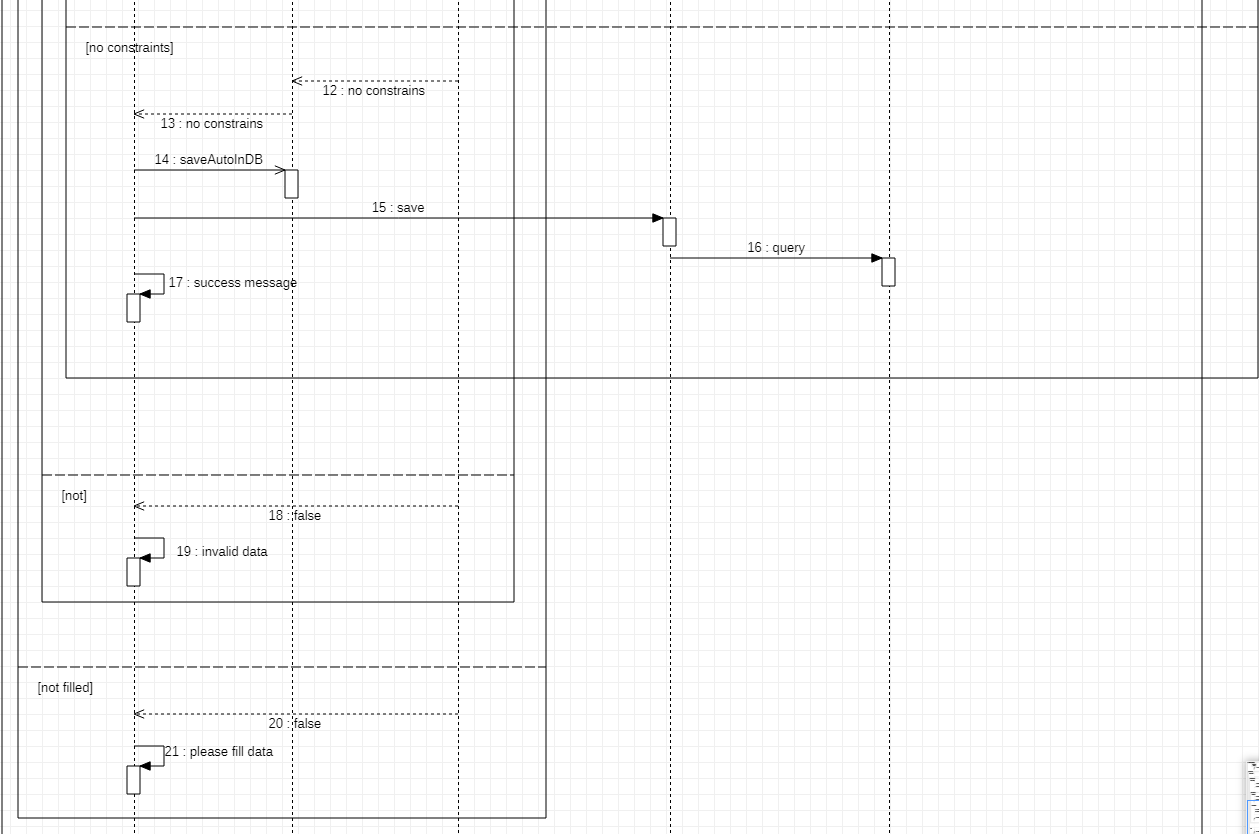
1. Front-end Class Diagram



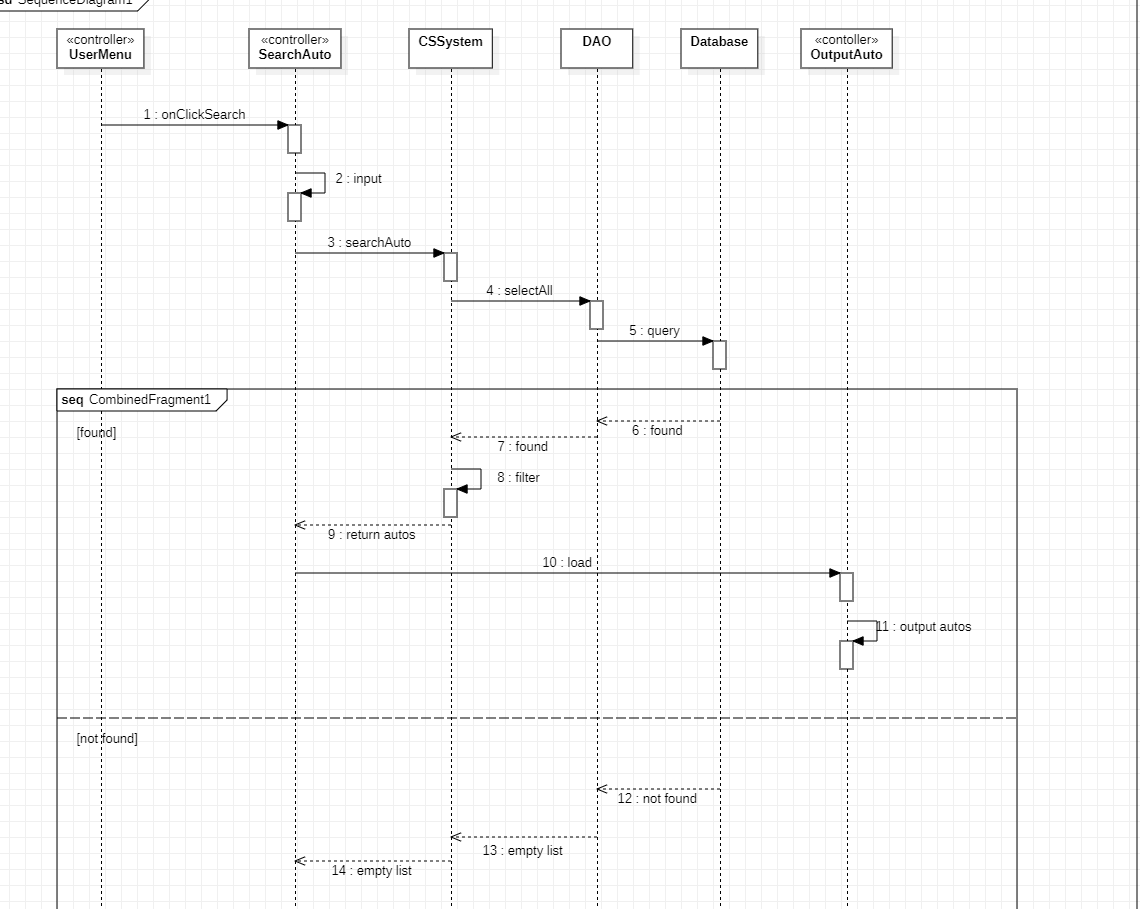
1. Sequence Diagram for Admin Login



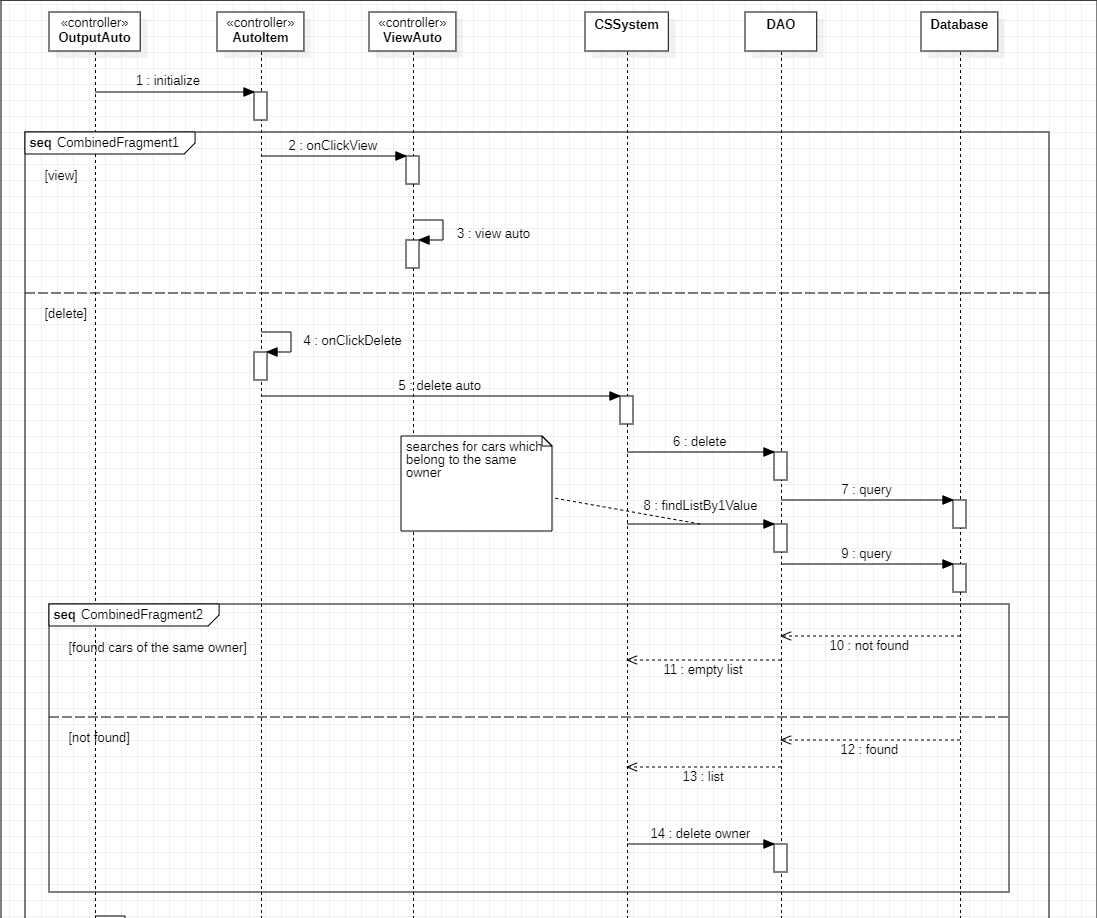
1. Sequence Diagram for adding automobile



Sequence Diagram for searching automobiles



1. Sequence Diagram for viewing or deleting auto



Изводи

1. Използването на система за управление на проекти значително помага за следенето на работата на участниците в екипа, разпределяне на задачите и улеснява управлението на целия софтуерен проект.
2. Иползването на система за контрол на версиите значително подобри комуникирането и работата в екип, тъй като дава възможност за изпълняване на работата едновременно и следене за конфликти