Съдържание

[1. Увод 2](#_Toc484692610)

[1.1. Актуалност на проблема и мотивация 2](#_Toc484692611)

[1.2. Цел и задачи на дипломната работа 3](#_Toc484692612)

[1.3. Очаквани ползи от реализацията 4](#_Toc484692613)

[1.4. Структура на дипломната работа 4](#_Toc484692614)

[2. Обзор на предметната област 5](#_Toc484692615)

[2.1. Методи за организиране на пътешествия 5](#_Toc484692616)

[2.1.1. Запис на хартиен носител 5](#_Toc484692617)

[2.1.2. Запис на дестинации в електронен документ 7](#_Toc484692618)

[2.1.3. Използване на софтуерно приложение (уебсайт) 8](#_Toc484692619)

[2.2. Съществуващи софтуерни решения 9](#_Toc484692620)

[2.2.1. TripAdvisor 9](#_Toc484692621)

[2.2.2. Opoznai.bg 11](#_Toc484692622)

[2.2.3. Sygic Travel 14](#_Toc484692623)

[2.3. Сравнителен анализ на решенията 16](#_Toc484692624)

[2.4. Изводи 17](#_Toc484692625)

[3. Използвани технологии и методологии 18](#_Toc484692626)

[3.1. Изисквания към средствата (технологии, методологии) 18](#_Toc484692627)

[3.1.1. Видове методологии 21](#_Toc484692628)

[3.2. Избор на средствата (технологии и методологии) за изпълнение на проекта 27](#_Toc484692629)

[3.2.1. Избор на методология 27](#_Toc484692630)

[3.2.2. Технологии 29](#_Toc484692631)

[Най-често използваните HTTP методи са: 32](#_Toc484692632)

[4. Анализ 38](#_Toc484692633)

[4.1. Концептуален модел 38](#_Toc484692634)

[4.2. Функционални изисквания 39](#_Toc484692635)

[4.2.1. Диаграма на потребителските случаи (Use case diagram) 40](#_Toc484692636)

[4.3. Нефункционални изисквания 44](#_Toc484692637)

[4.3.1. Използваемост 45](#_Toc484692638)

[4.3.2. Изправност 46](#_Toc484692639)

[4.3.3. Модулност и изменяемост 46](#_Toc484692640)

# Увод

## Актуалност на проблема и мотивация

Казват, че човек трябва поне веднъж годишно да посещава места, на които никога не е бил.

Смята се, че харченето на средства за опит е по-вероятно да донесе трайно щастие, отколкото тяхното прахосване по материални неща. Причината е, че хората се адаптират към физическите обекти - което означава, че нещата, които са купили с течение на времето ще им носят все по-малко радост. Обратно на това, радостта от преживяно пътуване и емоцията от спомените само се увеличава.

Пътуването е свързано предимно и само с положителна емоция. И за да се пътува, не винаги се изисква много, дори разходката до близкия град, планина или музей са пътешествие и нови емоции.

След като вече е набелязано как и къде да се прекара вободното време, остава едно от най-важните неща - организирането на пътуването. Със сигурност някои хора не се доверяват на организаторските си способности и предпочитат агенция да организира всичко - местата, които да посетят, хотелите в които да отседнат и т.н.

Всъщност в ерата на развитите информационни технологии самостоятелното организиране на пътуване е напълно възможно, а минусите от евентуално попадане сред група екскурзианти с разнопосочни интереси, интелект, култура и предпочитания са много. В повечето случаи се спестяват и разходи, но по-важното е, че пътувайки самостоятелно, хората се застраховат от редица разочарования като например да стигнат до мечтаната дестинация, без да видят всичко, което интересува лично тях, да чакат другите екскурзианти да се съберат навреме за обзорната обиколка на града или да разгледат най-любопитната според тях местна забележителност за 15 минути, вместо за цял ден.

Най-големият плюс на самостоятелно организираното пътуване е възможността да се създадат личен уникален маршрут и преживявания, с които истински да се усети магията на пътешествието.

## Цел и задачи на дипломната работа

Целта на дипломната работа е да се проектира и разработи уебсайт – „Trip Planner“, който ще се използва за планиране на маршрути за пътешествия. „Trip Planner“ ще представлява удобен помощник за построяването на маршрут за следващи пътувания. Потребителят ще може да запазва места, които иска да посети в бъдеще като ще добавя точното им местоположение на географска карта. Допълнително ще може да добавя снимка и информация за всяко място. Когато му предстои пътуване, ще може да види всички запазени места, които са по маршута до желаната дестинация, вместо да губи време да ги търси отново.

Друга полезна функция на уебсайта ще бъде филтрирането на места в близост до дадена греографска локация по зададено време или разстояние.

Задачите произтичащи от целта са:

* Специфициране на изискванията и проектиране на приложението;
* Представяне на използваните технологии и архитектурни решения;
* Сравнителен анализ на съществуващи софтуерни приложения с подобна функционалност;
* Разработване на отделните модули и интерфейс на приложението:
* Реализиране на възможност потребителите да запазват, редактират и разглеждат различни локации върху географска карта;
* Разработване на модул за работа с база от данни;
* Реализиране на възможност потребителите да построяват и запазват маршрут между отделни локации на картата;
* Реализиране на възможност потребителите да виждат запазените локации, които са в близост до зададения маршрут, като сами определят разстоянието или времето, необходимо да се достигнат;
* Реализиране на помощен интерфейс с напътстващи съобщения;

## Очаквани ползи от реализацията

“Trip Planner” ще помогне на потребителите лесно да съхраняват и разглеждат желаните дестинации. Основното преимущество на уебсайта ще бъде изобразяването на запазените места върху географска карта, което ще помага за по-глобален и систематизиран поглед върху тях. Избирайки маршрут, потребителят ще може много лесно да види места, които преди време е харесал, и ще може да ги включи към екскурзията си. Опцията за избор колко точно отдалечени да са те, ще помага за правилно планиране на времето. При смяна на снимката на мястото, ще се сменя и маркерът, който изобразява неговата позиция върху картата, което ще помага за разпознаване на запазената дестинация още от пръв поглед. Ще има и възможност за запазване на повече информация за мястото, която ще се показва само след кликване върху маркера.

Ще има възможност за построяване на маршрут между повече от две географски точки, както и опция за запазване на вече построен такъв. Веднага след изобразяването на маршрут ще се визуализират тези от запазените места, които са отдалечени според желанията на потребителя.

Тъй като потребителят може да не желае да посещава места, разположени по целия маршрут, ще има вариант за задаване само на една географска точка, при което ще се филтрират и покажат всички места в зададения радиус около нея.

Друга полза ще бъде възможността да се използва сайтът и на мобилни устройства. Потребителят ще може да построява или разглежда вече построени маршрути много лесно и удобно, дори когато вече е започнал своето пътешествие и е на път.

## Структура на дипломната работа

?????

# Обзор на предметната област

Със сигурност самостоятелното организиране на пътуване не е лесна задача, но ползите от него са много. Отделяйки време да планират пътуването си, хората не само спестяват много парични разходи, но и могат да са сигурни, че ще прекарат ваканцията си по свой вкус и предпочитания, без да се съобразяват с други пътешественици. Освен това самата организация и избиране на местата, които да се посетят е част от емоцията на цялото пътуване и зарежда с приятни чувства.

В повечето случаи избраната дестинация не е случайна. Тя може да е препоръчана от познати, човек може да е прочел статия за нея в интернет или списание, или дори просто да е видял снимка в социалните медии. Често хората харесват дадени места и искат да ги посетят, но след това забравят за част от тях. Или ако не забравят, не очакват, че ако се отклонят малко от маршрута за следващата си почивка, могат да посетят място, което са копнеели да видят дълго време.

## Методи за организиране на пътешествия

Следващите точки демонстрират предимствата и недостатъците на различните методи за предварително запазване на желани дестинации и информация за тях, и построяване на маршрут за следващо пътуване, включващ част от дестинациите, които са в близост и е оправдано да се посетят.

### Запис на хартиен носител

Човек може да записва местата, които иска да посети в тетрадка или тефтер, или пък дори да ги принтира от компютър и да ги организира в папка. Друга полезна тактика е използването на така наречените sticky notes. Това са малки листчета хартия, които в горния си край имат леко лепило. Харесаните дестинации могат да се записват на такива листчета и да се залепват върху голяма географска карта на стена, като се поставят точно на географската си локация. Вместо sticky notes, върху картата могат да се поставят принтирани снимки на дестинациите, които да се закрепват с помощта на габърчета.

**Предимствата на този метод са следните:**

* Използването на тефтер или тетрадка спомага за лесна преносимост;
* Взирането върху хартиен носител не натоварва очите толкова, колкото върху компютърен екран;
* Използването на географска карта и sticky notes или снимки помага за по-ясен поглед къде точно се намират местата;
* Използването на хартиена географска карта, окачена на стена, е по-забавно и могат да участват повече хора;
* Използването на sticky notes е гъвкаво, защото при грешка лесно може да се замени едно листче с друго;
* Може да се добавя информация с различен произход, без да се съобразява това с някакви ограничения, които биха съществували в софтуерно приложение;

**Недостатъците на метода са:**

* Използването на тефтер или тетрадка не е гъвкаво, защото трудно могат да се правят промени;
* При използването на тефтер или организирането в папка липсва ясната представа къде точно се намира мястото;
* Трудно се определя колко точно е отдалечено мястото от маршрута, дори с използването на географска карта;
* Използването на географска карта и sticky notes ограничава количеството на местата, които могат да се добавят. Добавянето на твърде много места или такива с близка локация води до претрупване и липса на ясно разбиране.
* Използването на географска карта изисква наличие на свободна стена, където тя да бъде окачена;
* За организирането на пътешествия в целия свят може да са необходими повече карти. Използването на световна карта няма да е удобно, тъй като тя ще е с много голям мащаб;
* Географска карта със закачени по нея sticky notes или снимки, не е преносима;
* Купуването на географски карти не е бюджетен вариант;
* Използването на хартия прави метода неприродосъобразен;

### Запис на дестинации в електронен документ

Харесани дестинации могат да се записват в електронни документи като могат да се добавят техни снимки и информация.

**Предимствата на този метод са следните:**

* Лесно и бързо създаване и съхранение;
* Възможност за копиране на информация и сваляне на снимки от интернет;
* Лесно редактиране;
* Възможност за проследяване на направените редакции;
* Възможност за групиране на дестинациите по папки;
* Възможност за споделено използване на файловете от много хора на различни географски локации;
* Възможност за контрол кой да вижда файловете и кой да може да ги редактира;
* Лесна преносимост. Може да се използва дори телефон;
* Безплатно;

**Недостатъците на метода "Запис в електронен документ" са:**

* Търсенето на места, които са близки до точки от маршрут, е много трудно и могат да се изпуснат желани дестинации;
* Трудно е да се прецени къде точно е географската локация на мястото;
* Трудно е да се прецени колко точно отдалечена е една точка от друга;
* При нарастването на обема на запазените дестинации, търсенето между тях става все по-трудно;
* Информацията е достъпна само чрез използване на съответния софтуер, а понякога различните софтуерни решения не са съвместими помежду си;
* Зависимост от хардуера – батерията може да падне или дори устройството може да се счупи в неподходящ момент;

### Използване на софтуерно приложение (уебсайт)

След електронните файлове, софтуерните приложения се появяват като още по-удобен начин за обработка. Те предлагат много по-голяма функционалност и са пълни с улеснения. Съществуват както уеб, така и десктоп решения. Те покриват различни типове устройства - от персоналния компютър до мобилния телефон.

**Предимства на метода "Софтуерно приложение":**

* Най-гъвкавият метод. Може да се използва на различни платформи и устройства;
* Удобен интерфейс;
* Спестява време - въвеждането става много по-удобно и бързо;
* Предоставя много полезни и разнообразни функционалности;
* Използването на уеб приложение дава възможност за достъп до информацията от различни устройства без нужда от инсталиране на софтуер;

**Недостатъци на метода "Софтуерно приложение":**

* Може да изисква такса за месечна/годишна поддръжка;
* Ако е използван друг метод преди преминаването към използване на софтуерно приложение, може да не е много лесно да се прехвърли информацията и е вероятно да отнеме време;
* За някои приложения може да са нужни по-задълбочени познания в сферата на компютърната грамотност;
* Използването на уеб приложение изисква достъп до интернет;
* Зависимост от хардуера – батерията може да падне или дори устройството може да се счупи в неподходящ момент;

## Съществуващи софтуерни решения

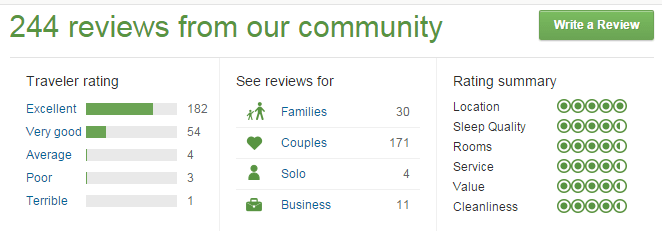
Съществуват много и разнообразни софтуерни решения за планиране на пътешествия, които предлагат разнообразни функции, като планиране посещението на препоръчани забележителности в даден град и дори запазване на самолетни билети и хотел. По-долу са разгледани най-успешните от тях.

### TripAdvisor

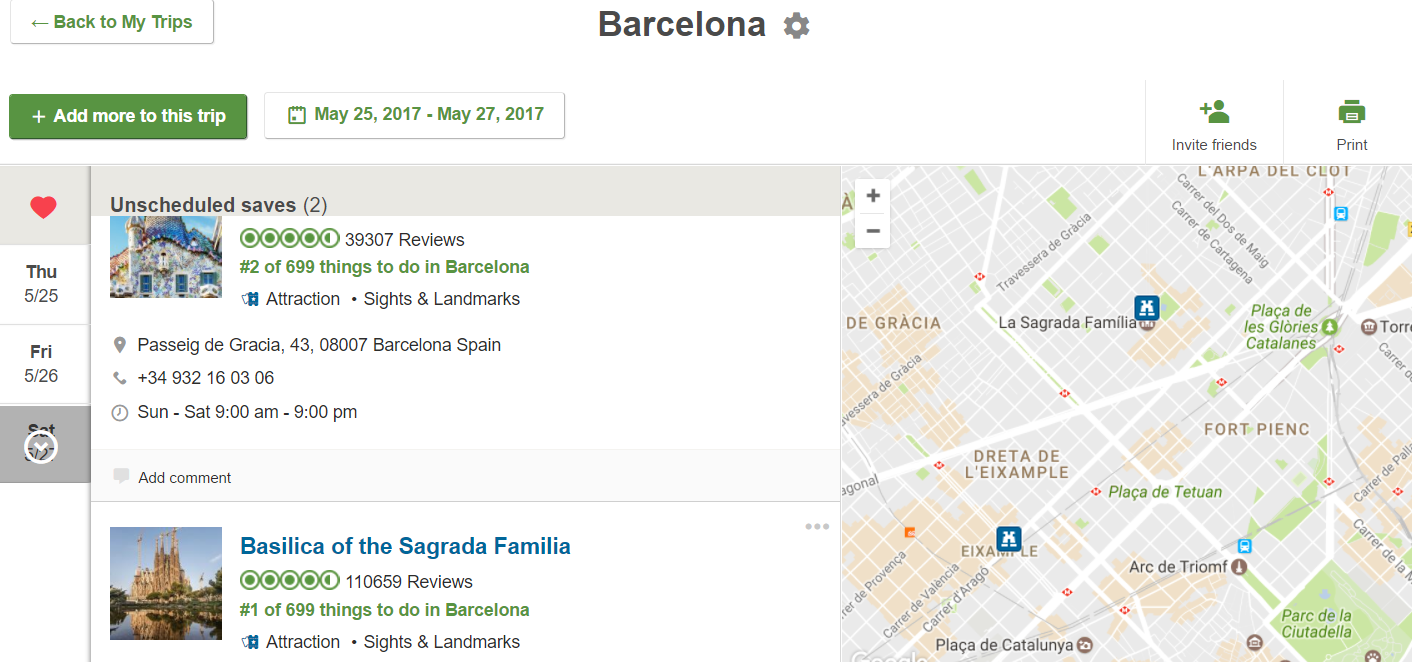
<https://www.tripadvisor.com/>

TripAdvisor е създаден през 2000 г. и днес представлява най-голямата колекция от ревюта на пътешественици за дестинации, хотели, ресторанти, забележителности, неща за правене и много други. След като дестинацията е избрана (или е налице колебание между няколко), доста лесно може да се прочете ценна и интересна информация за нея в TripAdvisor.

* В него може да се намерят и много удобни търсачки за хотели, апартаменти под наем и полети, които ще препратят към съответно най-изгодния сайт за резервация;
* Само с няколко клика могат да се намерят описания, класации и мнения на забележителностите, ресторантите и нещата, които могат да се правят в определен град или държава;
* Много бързо може да се разбере какво е качеството на услугата, която даден хотел предлага. TripAdvisor показва броя на гостите оставили ревюта, както и техния профил (“Семейства”, “Двойки”, “Индивидуални” или “Бизнес”). От тази информация може да се разбере дали хотела е подходящ за желаните цели (семейна почивка, романтичен уикенд, командировка от работа, и тн.) Също така, на първия панел има и илюстрирана  скала с броя на посетители, които са определили престоя си като съответно “Отличен”, “Много добър”, “Среден”, “Лош” или “Ужасен”, както и по-подробна разбивка на оценките (от 1 до 5) съответно за “Местоположението”, “Качество на спане”, “Стаи”, “Услуги”, “Стойност” и “Чистота”.



* Също така предоставя доста добре развити описания с мнения на посетители за “Нещата за правене” (Things to Do) в дадена дестинация. Информацията, включена в тази секция включва забележителности, турове, занимания, музеи, храна и напитки, транспорт, паркове, нощен живот, шопинг, театри, зоологически градини и много други. След като се избере една от тези категории, препоръките ще бъдат сортирани по популярност. Освен това, всяка категория има и подкатегории, които могат да се използват, ако се търси нещо по-конкретно.
* Предоставя възможност за предварително запазване на билети за посещение на обекти;
* Предоставя опция за създаване на списъци с места за посещение за дадено пътешествие. След това всички места, добавени в списък, могат да се видят изобразени върху географска карта.



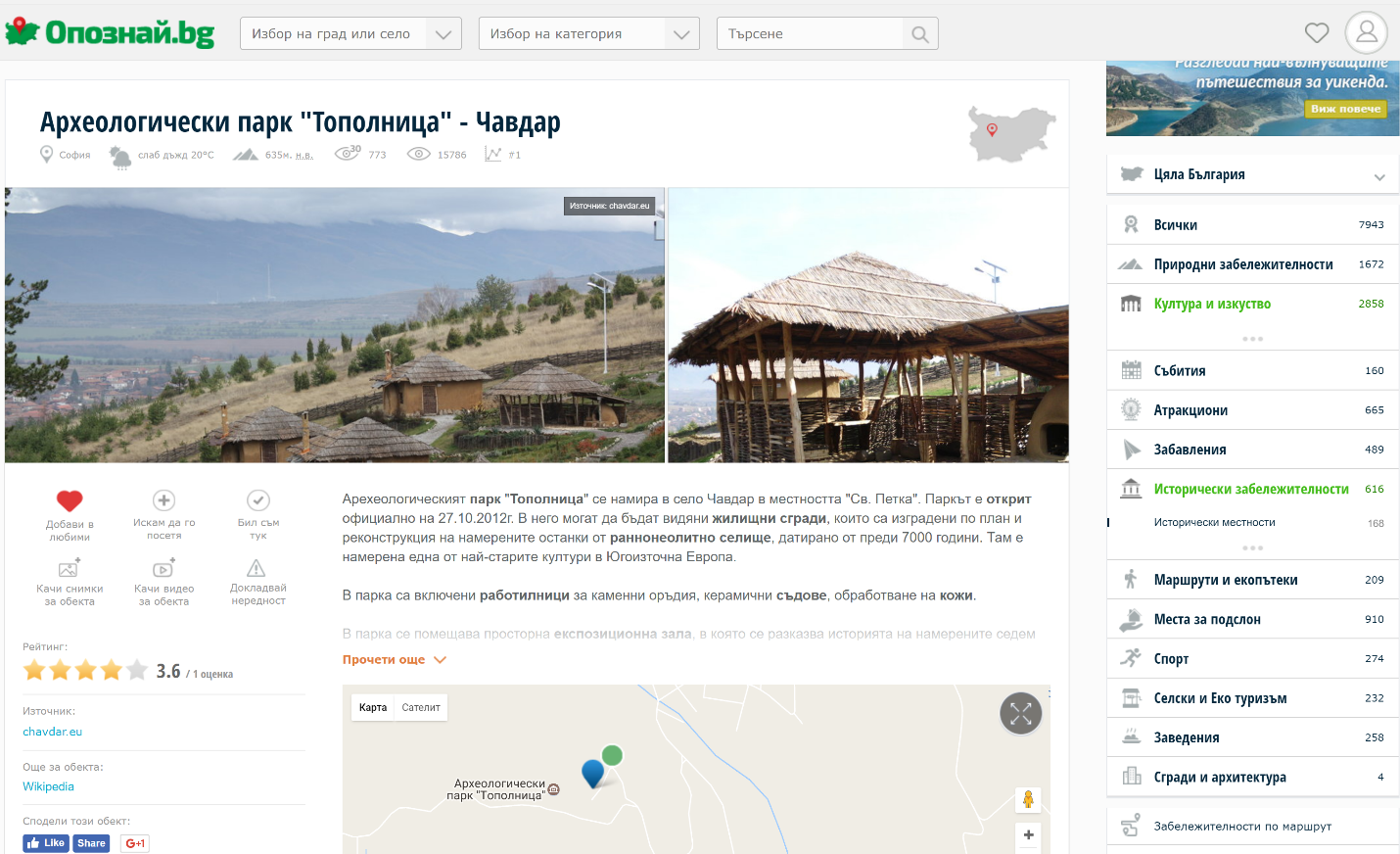
* Предлага списък с места, сходни до тези, които потребителят е разглеждал преди това;
* Може да се използва като уеб сайт или да се инсталира приложение на телефон;

### Opoznai.bg

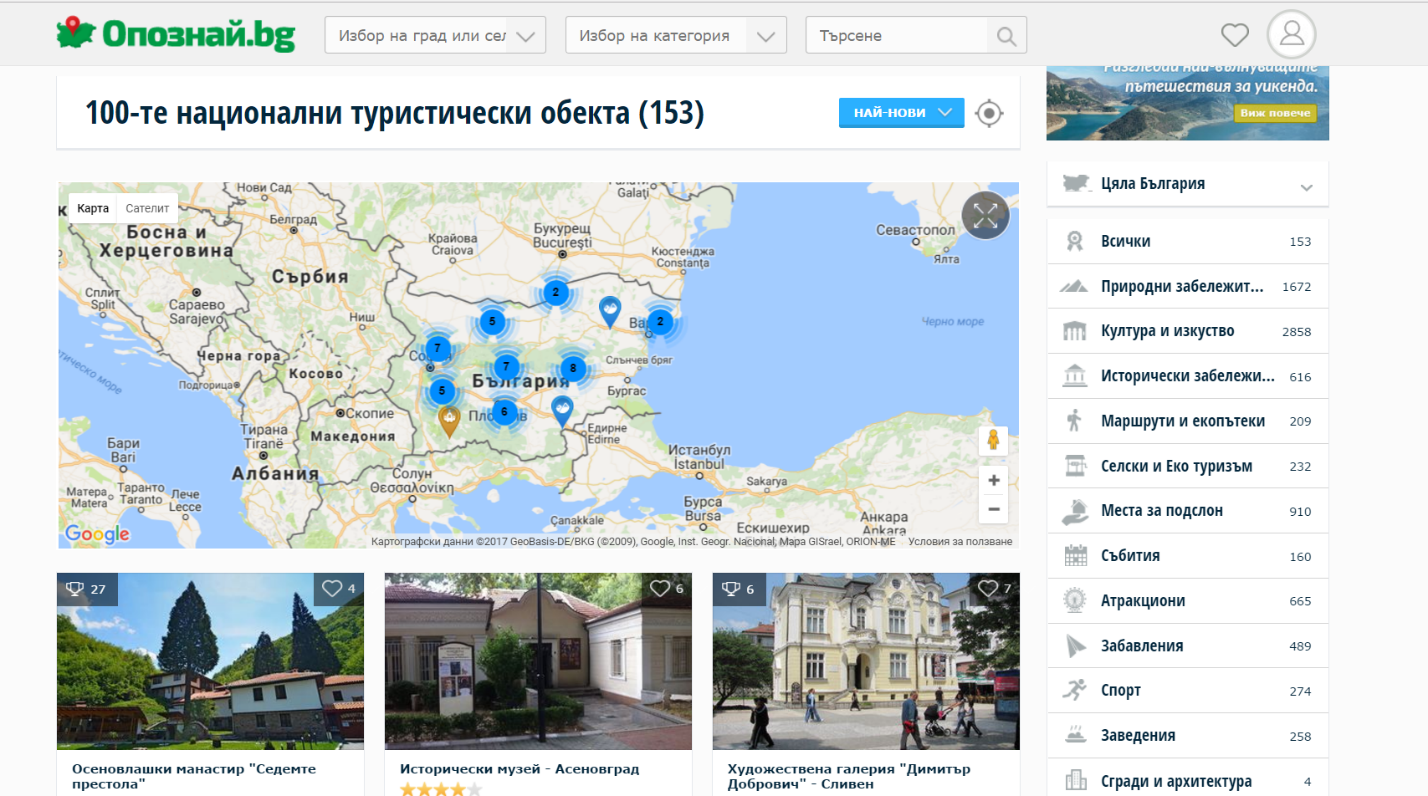
<https://opoznai.bg/>

Сайтът представлява пътеводител за България, който предлага богата колекция от забележителности и идеи за пътешествия, която непрекъснато се обновява от различни потребители.

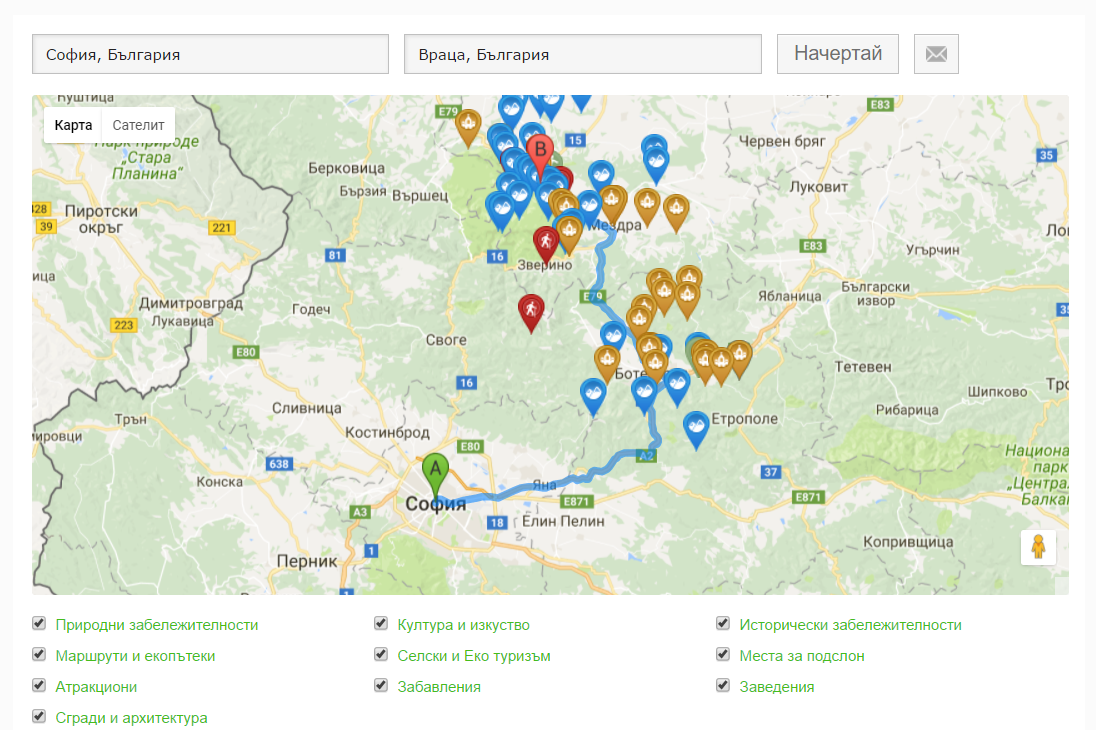
* Предлага възможност за избор на населено място или регион, при което се показват всички добавени забележителности в него. Показва се и списък с всички забележителности в близост;
* Местоположението на забележителностите се обозначава на географска карта;
* Забележителностите могат да се сортират по разнообразни критерии (най-посещавани, популярни, коментирани, търсени и др.)
* Забележителностите са разделени на различни категории и могат да се филтрират. Част от категориите са природни забележителности, културни, исторически, атракциони, забавления, места за подслон, заведения, сгради и архитектура, селски и еко туризъм, маршрути и екопътеки, събития и др.
* Потребителят може да добавя обектите, които е харесал, в списък с любими обекти;
* Всеки обект съдържа информация, снимки, точно географско местоположение, синоптична прогноза и коментари от потребители;



* Предлага списък с вече конструирани пътешествия в цяла България. Всяко пътешествие има период и обекти, които да се посетят;
* Предлага класации на най-популярните обекти;
* Предлага списък с текущи и предстоящи събития в цялата страна;
* Добавен е списък със 100-те национални исторически обекта;



* Предлага опция за търсене на забележителности по маршрут. При задаването на начална и крайна точка се изобразяват всички обекти по маршрута или в близост до него. Липсва опция за задаване на максимална отдалеченост на обектите; Има опция за филтриране на обектите по категория;

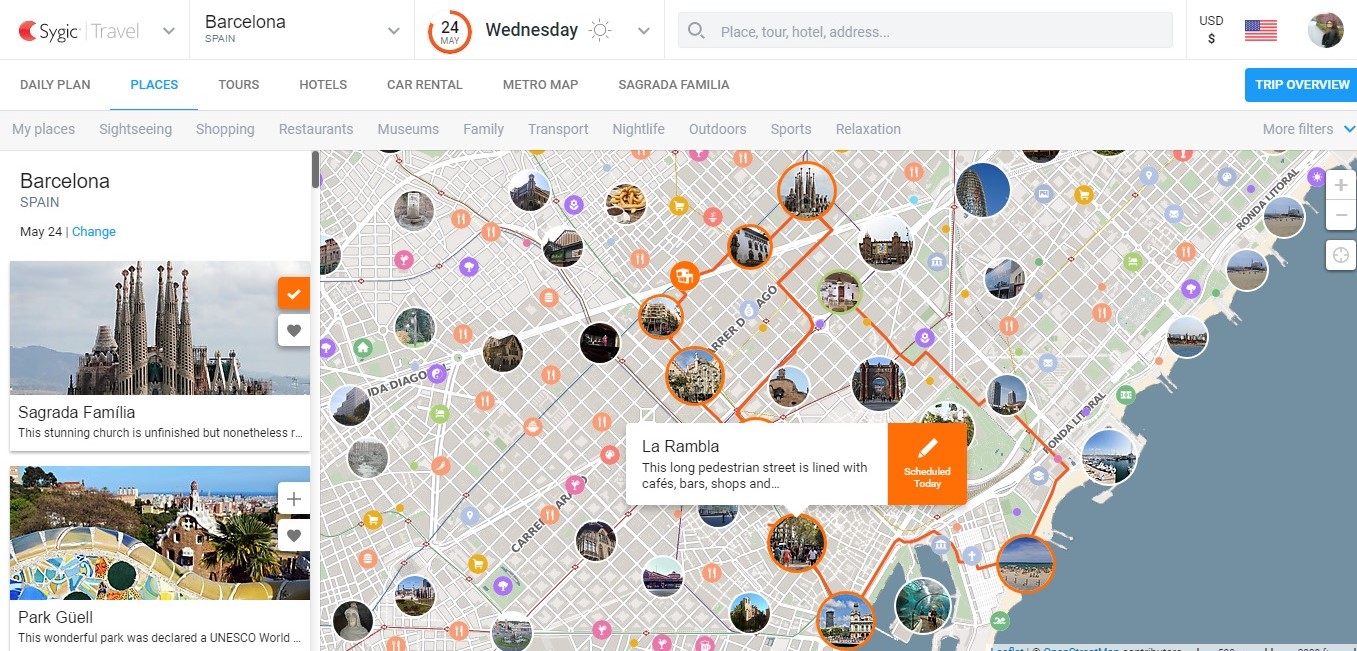


### Sygic Travel

<https://travel.sygic.com>

Представлява лесен за използване организатор за посещения на забележителности в даден град ден по ден.

* Още с отварянето на сайта се показва форма за въвеждане на дестинация на пътешествието и продължителност на престоя. На база на въведената информация приложението изготвя примерна програма за всеки ден, която може да се редактира;
* Локацията на забележителностите е изобразена на географска карта;
* При кликване на забележителност се показва информация за нея в панел отляво, която включва работно време, налични билети, турове и дори мнения на потребители;
* Местата могат да се филтрират по категории. Част от тях са: забележителности, шопинг, ресторанти, музеи, нощен живот, спорт, почивка и др. Всяка от категориите има и подкатегории. По такъв начин потребителят може да си изготви програма само с местата, които са по негов вкус;



* Сайтът предлага актуална карта на метрото на града и възможност за наемане на кола под наем;
* Друга полезна функция е търсачката на хотели. Намирането на място за престой е изключително лесно, тъй като всички предложения се обозначават върху карта и може да се избере най-удобното. Тук също са добавени възможности за филтрация, като част от тях са цена, звезди, ревюта, тип и др. Към филтрите могат да се добавят още много изисквания като наличие на паркинг, басейн, ресторант, безплатен интернет и др;
* Към едно пътешествие могат да се добавят повече от 1 градове, като избирането на забележителностите за всеки един става отделно. Панелът за навигация между градовете е много удобен;
* Всеки потребител има лист със всички пътешествия, които е планирал и запазил;
* Има възможност за покана на други потребители, които да участват в планирането на пътешествието;

## Сравнителен анализ на решенията

За сравнителен анализ ще се използват трите приложения от точка 2.2.

Най-важното изискване към едно приложение за планиране на пътешествия (и на всяко едно приложение като цяло) е то да е лесно и удобно за използване.

Наличието на приятен и интуитивен интерфейс спомага за превръщането на организацията на пътуване в един вълнуващ и интересен процес. Определянето на интерфейс като добър е доста субективно и индивидуално. За един потребител може да е по-интуитивен интерфейсът на TripAdvisor, за друг – този на Sygic. A макар нов и изцяло български сайт, opoznai.bg също може да се похвали с добър интерфейс. Освен това той има друго голямо предимство за хора, които не говорят чужди езици – сайтът е изцяло на български.

Друг критерий за използваемост е наличието на мобилна версия на сайта или приложението. Trip Advisor има приложение, налично за най-известните операционни системи – iOS, Android и Windows Phone. Към момента няма приложение на Sygic за Windows Phone, но мобилната версия на сайта е сравнително използваема и на мобилни устройства. Opoznai.bg също имат много удобно приложение за Android и iOS, а и мобилната версия на сайта е доста удобна и може безпроблемно да се използва на Windows Phone.

Да се направи функционално сравнение на всеки от трите софтуера не е лесно, тъй като всеки от тях предлага изключително голям набор от полезни инструменти и опции. Основните нужди за такъв тип сайт или приложение са покрити от всеки един от тях, но по различен начин. Въпрос на вкус и предпочитания е кой ще е най-удобен. В табл.1 са описани някои от основните изисквания при планиране на пътешествия и е обозначено дали са налице при всеки софтуер.

|  |  |  |  |
| --- | --- | --- | --- |
|  | TripAdvisor | Opoznai.bg | Sygic Travel |
| Планиране на пътешествия в цял свят | + | - | + |
| Списък с любими места | + | + | - |
| Търсене на забележителности в град | + | + | + |
| Търсене на забележителности по и около маршрут | - | + | - |
| Категории и филтри за забележителностите | + | + | + |
| Коментари и оценки от други потребители | + | + | + |
| Опция за изготвяне на програма (маршрут) | + | - | + |

( фиг. 1. Сравнителна таблица на съществуващи софтуерни решения )

Сайтът opoznai.bg е богат с много функционалности и информация, но е концентриран само в рамките на България, докато другите два могат да се използват за пътешествия в целия свят. Това е най-голямата разлика между тях. За сметка на това, той предлага много удобен инструмент за търсене на забележителности по и в близост до маршрут, който липсва при останалите. Всеки от трите софтуера поддържа търсене на забележителности в избран град, както и филтрацията им по категории. И докато в TripAdvisor и Sygic може да се изготви и запази програма от места за посещение в даден ден, то при българския вариант липсва такава опция. А тя е много удобна, защото показва всички добавени към програмата места върху географска карта и помага за по-глобален поглед и точна преценка. И трите софтуера предлагат възможност за споделяне на мнения и оценки за различни места от различни потребители.

## Изводи

Trip Planner ще бъде доста по-опростен помощник с много малко, но най-важните функционалности. Сайтът ще бъде концентриран върху две основни неща – запазване на места за по-нататъшно посещение и филтриране на тези от тях, които са в близост до дадена точка или маршрут. Това ще бъде неговото основно предназначение и няма да се добавя нищо излишно. Основната цел на Trip Planner е всичко да става изключително бързо и лесно, без излишни екрани и интерфейси.

Основната разлика на Trip Planner от разгледаните софтуери е, че той ще бъде личен. Ще съдържа и обработва само информация, добавена от потребителя. Няма да има препоръки, коментари, оценки от други хора или предварително добавени данни. Всичката информация, която ще се обработва ще се добавя от потребителя. Той ще вижда само места, които той е харесал и запазил, и маршрути, които лично е построил. Trip Planner ще представлява нещо като личен органайзер за дестинации за пътешествия.

Също така сайтът ще работи добре и на малки екрани, на мобилни телефони например, за да бъде възможно добавянето на нови места навсякъде и по всяко време.

# Използвани технологии и методологии

Следващата естествена стъпка след взимането на решение какво софтуерно приложение да се разработи, е изборът на технологии и методологии, които са най-подходящи за целта. Този избор зависи главно от изискванията, но не малка роля в него имат и опитът и компетентността на членовете на екипа.

Това е една от най-важните стъпки при стартирането на проект и тя не трябва да се подценява. Избирането на правилни технологии и методологии оказва огромно влияние върху производителността на екипа и спомага за спестяването на много средства и време.

Естествено ако не е направен правилен първоначален избор, дадена технология или методология може да се смени след като проектът вече е стартиран, но всяка непредвидена промяна има своята цена – обикновено продуктивността се намалява и се губи време. Обикновено колкото по-късно се правят големи промени на основни за проекта неща, толкова цената е по-висока.

## Изисквания към средствата (технологии, методологии)

При определяне на технологиите, които да бъдат използвани при разработката на едно софтуерно приложение много фактори играят роля. Възможно е при правенето на избор да се появят различни типове ограничителни условия - вътрешни убеждения или предпочитания на хората от разработващия екип, предпочитания на бъдещите клиенти, специфики на самото приложение, идеи за бъдещо развитие, финансови ограничения и други.

Някои от предимствата при правилен избор на технология са:

* По-добро планиране на сроковете, в които ще е готова всяка част от проекта;
* По-ниска цена за клиентите;
* По-бързо постигане на крайните резултати;
* По-малко усилия за разработчиците, а от друга страна по-малко бъгове;

Технологиите трябва да дават възможност да се изпълни поставената задача и да предоставят мощни помощни средства по време на разработката на софтуера. Други съществени изисквания са гъвкавост, цена, налични инструменти, популярност, среди за разработка, съвместимост. Вземането на решение на базата на критериите не е лесно, тъй като всяка технология има и позитиви, и негативи.

Следват съвети за правилно взимане на решение:

* На първо място е много хубаво да се разберат изискванията към крайният продукт. Какъв точно проблем трябва да се реши, какви цели трябва да се постигнат;
* Веднъж след като изискванията са събрани, те трябва да се разделят на логически бизнес модули. Ако даден модул е твърде неясен или абстрактен, той също трябва да се разбие на групи. От голяма полза би била диаграма на взаимодействията между отделните модули;
* След като изискванията са събрани и групирани, трябва да се отсеят основните – тези, от които зависи целият продукт. Така най-лесно ще се определи какъв тип ще е крайният продукт – дали ще е сайт за търговия, CMS (система за управление на съдържанието) система или др.
* Най-критичната част е проучването, защото при него се правят най-много грешки и се взимат неправилни решения.
  + След като основните модули са определени, трябва да се направи проучване в интернет и да се разгледат най-популярните технологии или решения;
  + Трябва да се изберат технологиите или решенията, които решават най-много от изискванията;
  + След като е избрана дадена технология, трябва да се направи проучване и за библиотеки, които да помагат за решаване на конкретния проблем;
  + Ако няма такива библиотеки, трябва да се направи планиране на необходимото време и усилия за разработване на решение в дадената технология;
  + Ако се окаже твърде трудно, трябва да се обмисли изборът на друга технология;

Методологията е набор от процедури, техники, средства заедно с документацията към тях, която подпомага разработчиците в усилията им да изградят нова информационна система.

Методологията се състои от [фази](http://tuj.asenevtsi.com/MetodA/MetodA02.htm), които  водят разработчика в неговия избор на техники, които са най-подходящи за всяка фаза на проекта и могат да го подпомогнат в планирането, управлението, контрола и оценката на проектите.

Предимствата от наличието на ясна и практична методология са много. Някои от тях са:

* Увеличаване на възможността за сравняване на различните проекти както и обмяната на ресурси между тях, благодарение на унифицираните определения със стандартизирани процеси за подпомагане на проекти;
* Поощряване повишаването на квалификацията чрез споделянето на добри практики в рамките на организацията, създавайки добри перспективи за кариера на участниците в проектните екипи, с което се осигурява и доброто бъдеще на компанията като цяло;
* Осигуряване успеха на проекта чрез по-добра подготовка, под формата на планиране, фокусирано на ключови моменти, още в първоначалната фаза на проекта;
* Ефективно взимане на решения. Когато са ясни ролите и очакваното поведение за участниците в проекта, вземането на решения е много по-ефективно. Смесването на отговорности е причина за забавяне на проекта;
* Обхватът на проекта се контролира. Ефективната методология помага за управлението на обхвата, който често е причина за превишени разходи и време;
* Клиентът знае какво да очаква. Често се случва крайният продукт да не отговаря на нуждите на клиента. Методологията за управление на проекти помага да се гарантира, че клиентът и разработващият екип са съгласни с това, което ще бъде доставено;
* По-добро разрешаване на проблемите. Самата методология изисква определянето на рискове още в началото на проекта и изготвяне на план за справяне с евентуално възникване на проблеми;

### Видове методологии

Едно от първите решения, които трябва да се вземат при стартирането на нов проект, е каква методология да се използва. Двете основни и най-популярни методологии са:

* Waterfall – традиционен подход;
* Agile – по-нов и гъвкав подход;

И двете методологии са полезни и широко използвани. В следващите точки ще се разгледат по по-подробно и ще се предостави сравнителен анализ на силните и слабите им страни.

#### Waterfall

Waterfall методологията се характеризира с линеен подход към разработването на софтуер. При нея последователността на събитията е точно определена и включва:

* Концепция;
* Иницииране;
* Анализ;
* Проектиране и дизайн;
* Разработване;
* Тестване;
* Внедряване;
* Поддръжка;

Преминаването към следващ етап настъпва само след като предходният е изцяло завършен, а връщането към предходен е недопустимо. Няма място за промени и грешки, затова очакваните резултати и планът за проекта трябва да бъдат точно зададени в началото и след това стриктно да се следват.

Като всяка методология, тази също има добри и лоши страни. Чат от добрите са:

* Разработчиците и клиентите се съгласяват какво ще бъде доставено в началото на жизнения цикъл на разработката. Това прави планирането и проектирането по-ясни;
* Напредъкът се измерва по-лесно, тъй като пълният обхват на проекта е известен предварително;
* Присъствието на клиентите не се изисква строго след изясняването на изискванията в началото;
* Тъй като проектирането и дизайнът на целия софтуер се извършва в началото, е много по-малко вероятно да има части от системата, които са неконсистентни с останалите;

Някои от недостатъците на методологията са:

* Тъй като всички резултати се основават на документирани изисквания, клиентът може да не види какво ще бъде доставено, докато не бъде почти готово. По това време промените могат да бъдат трудни (и дори невъзможни) за прилагане.
* Целият продукт се тества само в края. Ако бъговете са написани рано, но са открити късно, тяхното съществуване може да е повлияло на начина, по който е написан друг код.
* Ако клиентът осъзнае, че се нуждае от повече, отколкото първоначално е предвидил, и иска промяна, това ще се отрази на времето и бюджета.

Кога трябва да се използва Waterfall методология:

* Когато има ясна представа какъв трябва да е крайният продукт;
* Когато клиентите няма да имат възможност да променят обхвата на проекта след като вече е стартирал;
* Когато документацията е по-важна от скоростта;

#### Agile

Появява се като решение на недостатъците на Waterfall методологията. Agile е повтарящ се и изцяло осланящ се и зависещ от екипа подход за разработване на софтуер. При него се набляга на бързото предоставяне на приложение с вече работещи функционални компоненти.

Времето за разработване се разделя на така наречените спринтове. Всеки спринт има определена продължителност (обикновено от 1 до 4 седмици) и списък със задачи, определени в началото на спринта, които трябва да са готови и тествани в края му. Ако останат незавършени задачи, те се преместват в следващия спринт. В края на всеки спринт обикновено се прави демо на клиента със завършените функционалности. Agile разчита на много голямо участие на клиентите по време на разработването на целия проект.

Предимствата на методологията са:

* Тъй като клиентите взимат голямо участие в проекта, те могат да забележат нередности на много ранен етап от разработването и да поискат промени;
* Тъй като методологията позволява да се правят промени, е лесно да се добавят нови функции в крак с последните тенденции в дадената индустрия;
* Клиентът придобива силно чувство за собственост, тъй като работи директно с екипа по време на целия проект;
* Ако времето за пускане на пазара за конкретно приложение е по-важно от освобождаването на пълен набор от функции при първоначалното стартиране, Agile може по-бързо да произведе основна версия на работещ софтуер, който ще се надгради с повече функционалности на по-късен етап;
* Разработването е по-фокусирано върху нуждите на потребителите, заради получаването на постоянна обратна връзка от клиента;
* Тестването в края на всеки спринт гарантира, че бъговете се улавят и оправят на ранен етап;

Недостатъците са:

* Тъй като първоначалният проект няма окончателен план, крайният продукт може да бъде съвсем различен от първоначално предвидения.
* Много високата степен на участие на клиентите, макар и отлична за проекта, може да създаде проблеми за някои клиенти, които просто не могат да имат време или интерес за този вид участие.
* Участието на клиентите често води до допълнителни функции, поискани по време на проекта, което естествено води до увеличаване на цената и времето за разработване на проекта;
* Тесните работни отношения са най-лесни за управление, когато членовете на екипа се намират на едно и също физическо пространство, което не винаги е възможно. Има обаче различни начини за справяне с този проблем, като уеб камери, инструменти за сътрудничество и др.

Кога да се използва Agile:

* Когато е важно бързото пускане на пазара на части от функционалността на софтуера;
* Когато клиентите ще могат да променят обхвата на проекта;
* Когато няма ясна картина за това как трябва да изглежда крайният продукт;
* Когато продуктът е предназначен за индустрия с бързо променящи се стандарти;

#### Сравнителен анализ и избор между двете методологии

При избора на методология трябва да се отчетат следните фактори:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Agile | Waterfall | Коментари |
| Възможност за участие на клиента | Препоръчва се участие на клиента през целия жизнен цикъл на проекта | Изисква участие на клиента само при планиране и договаряне на изискванията в началото и на най-важните етапи от разработката. Като цяло тук участието е много по-малко. | Участието на клиента винаги намалява риска от разработването на функционалности, които не отговарят на нуждите на пазара или дори на желанията му. |
| Обхват/ функционалности | Работи с промени, но на съответната цена – време, средства или отстраняването на други функционалности. Работи най-добре, когато обхватът на проекта не се знае при стартирането му. | Работи най-добре, когато обхватът е уточнен в началото и договорът ограничава промени. | Необходимостта от промени е съвсем нормална, а понякога и неизбежна, затова е хубаво да се предпочитат по адаптивни и гъвкави методологии. Въпреки това, понякога договорите стриктно ги ограничават. |
| Приоритизиране на функционалностите | Приоритизирането по стойност гарантира, че най-важните функционалности ще бъдат разработени първо. Намалява се рискът от пълен провал, като постепенно се пускат на пазара успешни и работещи функционалности. | Накрая клиентът получава, всичко което е поискал в началото, но то може да не отговаря на текущите нужди на пазара. Рискът от провал е по-голям. | Договорът може да ограничава частичният успех. |
| Екип | Препоръчва се използването на малки, отдадени, синхронизирани и работещи заедно екипи. | Допуска се използването на различни екипи за различните части на системата. | Екипи, които работят заедно работят по-добре, но понякога се сключват договори с различни фирми за разработването на отделните части на софтуера. |
| Финансиране | Работи добре с нефиксирано финансиране, тъй като обхватът не е предварително и крайно договорен в началото на проекта. | Предварително договаряне на функционалности, които да се предоставят и фиксирано финансиране намаляват стреса и риска за екипа и фирмата. | Фиксираната цена е проблем, когато обхватът не се знае в началото, но много правителствени договори го изискват. |
| Изводи | Стига да е приложим, това е по-добрият вариант. | Когато има фиксирани изисквания и твърди договори, планирането и договарянето в началото намаляват риска от провал. Обикновено такива договори се сключват с правителствени организации. | В началото на всеки проект трябва подробно да се обясни на клиента всяка една от двете методологии и да се избере подходящата, като в повечето случаи по-удобната и предпочитана от екипа е Agile. |

Важно е да се отбележи, че горните фактори не са еднакво и еднозначно претеглени. Всеки от тях се оценява в зависимост от конкретния проект и обстоятелства.

## Избор на средствата (технологии и методологии) за изпълнение на проекта

### Избор на методология

За реализацията на уебсайта “Trip Planner” е използвана гъвкавата методология Scrum.

Тази методология е променила възприятията за типичното управление на проекти, като ясно показва предимствата на гъвкавите пред „waterfall“ или неитеративните, негъвкави методологии.

Спринтът е най-малката единица време за разработване. Спринтовете са с постоянна дължина от 1 седмица до 1 месец. Всеки спринт е опит за подобрение вкаран във фиксирани времеви рамки.

Преди всеки спринт има среща за планиране на спринта. На нея се поставят измеримите цели за спринта и се идентифицират задачите, които ще бъдат свършени в неговите рамки. По време на всеки спринт екипът създава завършени парчета от даден продукт. Задачите за всеки спринт се описват и взимат от хранилището със задачи или на английски „product backlog”. Често тези дейности са описани като характеристики, които продукта трябва да има и да бъдат постигнати за спринта. Т.е. product backlog е приоритизиран списък на изискванията. Какво от списъка да влезе в даден спринт се решава на планиращата среща преди спринта. Продуктовият собственик (Product Owner) уведомява екипа кои части от списъка с изисквания иска да бъдат свършени на предстоящия спринт. Екипът преглежда, обсъжда, решава и записва в Sprint Backlog кои от тези изисквания и цели ще успее да изпълни в рамките на предстоящия спринт. Sprint Backlog е собственост на екипа. Целите, вписани в този документ, не трябва да бъдат променяни по време на спринта. За разработването се определя фиксирана продължителност, такава, че спринтът да свърши навреме. Изискванията, които не бъдат удовлетворени за спринта се изключват и връщат към product backlog. След като спринтът е изпълнен, екипът демонстрира как се използва софтуерът.

Scrum позволява организирането на самоорганизиращи се екипи като стимулира всички членове на екипа да се намират на едно и също място и да си комуникират на живо.

Ключов принцип на Scrum е това, че се приема още в самото начало на проекта, че изискванията няма как да са пълни и напълно разбрани. Т.е. очакват се нововъведения от клиента – промяна на желанията му. Scrum се фокусира върху способността на екипа да доставя бързо и да е готов да отговори бързо на неочакваните промени. Това е положителна черта за Scrum, защото резките промени не могат да бъдат добре посрещнати с традиционните методи.

Ключови роли или самият SCRUM екип:

* Product Owner / Собственик на продукта - отговорното лице за това организацията да добавя стойност (към продукти и услуги) за клиентите. Той е гласът на клиента. Той общува с клиента, задава приоритети на задачите и вписва всичко в „product backlog”- а. В Scrum екипа такава роля трябва да има само един човек. Ролята на собственик не се препоръчва да се смесва с тази на Scrum ръководителя. От друга страна той може да бъде човек от екипа за разработка.
* Scrum Master / Scrum ръководител – той е отговорен да бъдат премахвани пречките при изпълнението на договорените в спринта задачи и да се постигнат желаните за спринта цели. Той не е лидер на екипа, а нещо като служещ на екипа лидер. Следи за изпълнение на правилата и за това нещата да се случват по концепциите на Scrum. Ръководителят се грижи за това екипът да не бъде разсейван със странични фактори и за това всичко да е подчинено на целите на спринта.
* Development Team / Екип разработка - екипът, който създава продукта, анализира, прави архитектурата, пише код, тества го, извършва работа по техническата комуникация, документира и т.н.. Той има задачата да доставя увеличен брой или подобрени парчета софтуер на края на всеки спринт. Тези парчета трябва да са по някакъв начин подходящи за използване и пускане на пазара. Така във всеки един момент проектът може да бъде спрян и продаден.

Основни понятия в методологията:

* Product Backlog  - както е упоменато по-горе, това е приоритетно подреден списък с изисквания, редактируем от всеки, но отговорност на продуктния собственик. При приоритизирането последния взема предвид рискове, добавена стойност, разходи, зависимости, дата на предаване на клиента и т. н.
* Sprint Backlog  - списък с работата, която екипът трябва да свърши. В нея по време на ежедневните Scrum срещи се вписват идентифицирани задачи. Те обаче не се задават на никого конкретно. Програмистите сами си избират задачи от там спрямо приоритети.

По време на спринта, всеки ден се провеждат т.нар. правостоящи срещи (stand-up meetings). Тези срещи продължават от 5 до 15 минути и се провеждат ежедневно в определен час. На срещата всеки от екипа абсолютно неформално разказва за три неща:

* какво е работил предишния ден;
* какво планира за предстоящия ден;
* какви проблеми е срещнал, които му пречат да работи;

Главни атрибути на срещата са:

* Срещата започва точно навреме;
* Мястото и часът са едни и същи всеки ден;
* Срещата трябва да се вмести точно в 15 минути;
* Може всички, но главно говорят основните роли в екипа;

На срещата се обновява и backlog-ът като се отбелязва свършената работа.

Ако се идентифицират някакви проблеми, те се решават колективно. Важно е да се отбележи, че това не са срещи за отчет пред ръководството, а за синхронизация (самоорганизация) на екипа и разкриване на потенциални пречки в работата.

### Технологии

#### Език за програмиране

Разработването на уебсайта ще се извърши на езика Javascript.

JavaScript е програмен език, който позволява динамична промяна на поведението на [браузъра](https://bg.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D1%8A%D1%80) в рамките на дадена [HTML](https://bg.wikipedia.org/wiki/HTML) страницата. JavaScript се зарежда, интерпретира и изпълнява от уеб браузъра, който му осигурява достъп до [Обектния модел на браузъра](https://bg.wikipedia.org/wiki/DOM). JavaScript функции могат да се свържат със събития на страницата (например: движение/натискане на мишката, клавиатурата или елемент от страницата, и други потребителски действия). JavaScript е най-широко разпространеният език за програмиране в Интернет. Прието е JavaScript програмите да се наричат [скриптове](https://bg.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82).

#### База данни

За Trip Planner ще се използва MongoDB.

MongoDB е система за обработване на [бази данни](https://bg.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8) от документи, разработена от [10gen](https://bg.wikipedia.org/w/index.php?title=10gen&action=edit&redlink=1). Тя е от рода на [нерелационните бази данни (NoSQL)](https://bg.wikipedia.org/wiki/NoSQL). Вместо да съхранява информация в таблици, както е при традиционните [релационни бази данни](https://bg.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8), MongoDB съхранява структурираната информация в [JSON](https://bg.wikipedia.org/wiki/JSON) формат с динамични схеми. Това прави интегрирането на информацията в определени приложения доста по-лесно и по-бързо.

Нерелационната база данни (на [английски](https://bg.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8_%D0%B5%D0%B7%D0%B8%D0%BA): Not only Structured Query Language, NoSQL) предоставя механизъм за съхранение и възстановяване на данни, който използва свободен [съгласуван модел](https://bg.wikipedia.org/w/index.php?title=%D0%A1%D1%8A%D0%B3%D0%BB%D0%B0%D1%81%D1%83%D0%B2%D0%B0%D0%BD_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB&action=edit&redlink=1) за разлика от по-често ползваната [релационна база данни](https://bg.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8). Ползите на този подход включват изчистен дизайн, [хоризонтално мащабиране](https://bg.wikipedia.org/w/index.php?title=%D0%A5%D0%BE%D1%80%D0%B8%D0%B7%D0%BE%D0%BD%D1%82%D0%B0%D0%BB%D0%BD%D0%BE_%D0%BC%D0%B0%D1%89%D0%B0%D0%B1%D0%B8%D1%80%D0%B0%D0%BD%D0%B5&action=edit&redlink=1) и фин контрол върху наличнaта информация. Нерелационната база данни е най-често добре оптимизирано хранилище, съдържащо информация от тип [ключ-стойност](https://bg.wikipedia.org/w/index.php?title=%D0%9A%D0%BB%D1%8E%D1%87-%D1%81%D1%82%D0%BE%D0%B9%D0%BD%D0%BE%D1%81%D1%82_%D1%85%D1%80%D0%B0%D0%BD%D0%B8%D0%BB%D0%B8%D1%89%D0%B5&action=edit&redlink=1). Предназначението ѝ е да улесни процесите по възстановяване и добавяне на информация, с цел оптимизиране на производителността в условия на неумишлено забавяне на системата и въвеждане на прекалено големи количества данни.

Когато са въведени релационните бази данни в 70-те години на миналия век, схемите за данни са били доста прости и ясни и е имало смисъл да се построяват обекти като съвкупности от връзки. Например обектът статия може да е свързан с категория (обект), таг (друг обект), коментар (друг обект) и т.н.

Тъй като връзките между различните типове данни са били посочени в схемата на базата данни, те са могли да се извличат със стандартен език за структурирани заявки или SQL. Но средата за данни, както и програмирането, се промениха в последните години:

* Появата на изчислителни облаци доведе до драматично намаляване на разходите за разполагане и съхранение, но само ако данните могат да се разпространяват лесно на няколко сървъра без да се нарушават. В сложна SQL база данни, това е трудно, защото много заявки изискват множество големи таблици да бъдат свързани, за да се даде отговор. Изпълнението на разпределени връзки е много сложен проблем в релационните бази данни.
* Необходимостта от съхраняване на неструктурирани данни, като публикации в социалните медии и мултимедия, нарасна бързо. SQL базите данни са изключително ефикасни при съхраняването на структурирана информация , но са необходими заобикаляне или компромиси за съхраняване и извличане на неструктурирани данни.
* Популярните Agile методологии изискват схемата на базата данни бързо да се променя, тъй като изискванията се развиват. SQL базите данни изискват тяхната структура да бъде предварително определена, което означава, че всяка промяна в информационната схема изисква отнемащите време операции.

В отговор на тези промени се появиха нови начини за съхранение на данни, които позволяват данните да бъдат групирани по-естествено и логично и да се разхлабят ограниченията върху схемата на базата данни. Един от най-популярните начини за съхранение на данни е моделът на документа, където всеки запис и свързаните с него данни се смятат за "документ". В базата данни на документи, като MongoDB, всичко свързано с обект на база данни е капсулирано заедно. Съхраняването на данни по този начин има следните предимства:

* Документите са независими единици, което прави производителността по-добра и улеснява разпространението на данни между множество сървъри;
* Логиката на приложението е по-лесна за писане. Не е необходимо обектите от заявката да се превеждат на език, удобен за разработване и обратното;
* Неструктурираните данни могат лесно да се съхраняват, тъй като документът съдържа всички ключове и стойности, които изисква логиката на приложението. Освен това се избягват скъпи миграции, тъй като базата данни не се нуждае от предварителна информация за своята информационна схема;

Базите данни, базирани на документи обикновено имат много мощни заявки и индексиращи функции, които улесняват и могат бързо да изпълняват много различни оптимизирани заявки.

MongoDB e документно хранилище. Централната концепция, която стои зад документното хранилище е нотацията за „документ“. Докато всяка документно-ориентирана имплементация се различава според детайлите на тази дефиниция, като цяло, всички те приемат, че документите енкапсулират и кодират данните (или информацията) в някакви стандартни формати. Използваните формати са: [XML](https://bg.wikipedia.org/wiki/XML), [YAML](https://bg.wikipedia.org/w/index.php?title=YAML&action=edit&redlink=1), [JSON](https://bg.wikipedia.org/wiki/JSON) както и бинарни формати като [BSON](https://bg.wikipedia.org/w/index.php?title=BSON&action=edit&redlink=1), [PDF](https://bg.wikipedia.org/wiki/PDF) and [Microsoft Office](https://bg.wikipedia.org/wiki/Microsoft_Office) документи ([MS Word](https://bg.wikipedia.org/wiki/MS_Word), [Excel](https://bg.wikipedia.org/wiki/Excel) и др.).

Различните имплементации предлагат различен подход към организирането и групираненето на документите:

* Колекции
* Тагове
* Невидими метаданни
* Йерархия на директориите

Ако се направи аналогия с релационните бази данни, колекциите могат да се възприемат като таблици, а документите могат да се приемат за записи. Нерелационните и релационните база данни все пак има съществена разлика: всеки запис в таблицата при релационните бази данни има еднаква последователност от полета, докато документите в колекцията могат да имат полета, които са напълно различни.

Адресът на документите се представя в базата данни чрез уникален ключ, който идентифицира документа. Една от другите основни характеристики на документно-ориентираните бази данни е, че освен простото търсене по ключ-документ или ключ-стойност, което може да се използва за извличане на документа, базите данни предоставят и потребителски интерфейс или език за заявки, който ще позволи документите да бъдат откривани въз основа на тяхното съдържание.

С нарастването на популярността на MongoDB се появяват различни варианти за отдалечено хостване. За Trip Planner ще се използва MLab.

MLab е услуга за съхранение и управление на база данни в облак. Има възможност за създаване на безплатен акаунт, който предоставя 500 MB пространство. Всеки mLab акаунт предоставя и Data API (като RESTful интерфейс през HTTPS), което може да се използва, за да се достъпи базата, колекциите и документите, принадлежащи на дадения акаунт. Всеки потребител има свой собствен API ключ, който може да се използва за достъп до приложния програмен интерфейс (API). Потребителите могат да преглеждат или нулират своите API ключове чрез портала за управление. Всеки API ключ осигурява пълен достъп до всички данни в базите данни, принадлежащи дадения акаунт.

RESTful уеб API (също наричано RESTful уеб service) е уеб приложение,което използва принципите на HTTP и REST. Представлява колекция от ресурси със четири дефинирани аспектa:

* Основният "URL" за уеб приложенията като <http://example.com/resources/>;
* [Internet media](http://en.wikipedia.org/wiki/Internet_media_type) типът на данните поддържани от уеб приложенията. Това най-честo е [JSON](http://en.wikipedia.org/wiki/JSON), но може да бъде всеки друг валиден Интернет медиен тип, като се има предвид, че е валиден хипертекст стандарт;
* Операции поддържани от уеб приложението използвайки [HTTP методи](http://en.wikipedia.org/wiki/Request_method) (примерно: GET, PUT, POST, или DELETE);
* Приложенията трябва да се задвижват от хипертекст;

## Най-често използваните HTTP методи са:

* GET - Осигурява достъп само за четене до ресурс;
* PUT - Използва се за създаване на нов ресурс;
* DELETE - Използва се за отстраняване на ресурс;
* POST - Използва се за актуализиране на съществуващ ресурс или за създаване на нов ресурс;

Методите, които предоставя mLab API-то са:

* List databases (GET) – връща всички бази данни на потребителя;
* List collections (GET) – връща всички колекции в дадена база;
* List documents (GET) – връща всички документи в дадена колекция;
* Insert document (POST) – създава нов документ в дадена колекция;
* Insert multiple documents (POST) – добавя няколко документа към дадена колекция;
* Update multiple documents (PUT) – променя дадени документи в колекция;
* Delete/replace multiple documents (PUT) – изтрива или заменя дадено документи;
* View a single document (GET) – връща документ с даден идентификатор;
* Update a single document (PUT) – променя документ по даден идентификатор;
* Delete a single document (DELETE) – изтрива документ с даден идентификатор;

#### jQuery

[jQuery](http://bg.wikipedia.org/wiki/JQuery) е една от най-известните и най-използваните библиотеки на JavaScript. Нейният създател е [Джон Резиг](http://bg.wikipedia.org/wiki/%D0%94%D0%B6%D0%BE%D0%BD_%D0%A0%D0%B5%D0%B7%D0%B8%D0%B3" \t "_blank" \o "Джон Резиг в wikipedia), а през 2006 година е пусната официално в Интернет.

Библиотеката е проектирана да улесни и опрости писането на JavaScript код, като едновременно с това е гъвкава и може да се надгражда с допълнителни плъгини.

Библиотеката се [поддържа от всички съвременни версии на уеб браузърите](http://jquery.com/browser-support" \t "_blank" \o "jQuery Current Active Support).

jQuery e безплатен софтуер с отворен код под MIT и GNU GPL лицензи. Синтаксисът на jQuery позволява лесната и бърза работа с DOM (Document Object Model) елементи, улеснява създаването на анимации взаимодействието с потребителя и позволява създаването на AJAX интернет приложения.

jQuery притежава следните възможности:

* Селеткиране на DOM елементи;
* Обхождане и модифициране на DOM дървото;
* Събития;
* CSS обработка;
* Ефекти и анимации;
* Ajax;
* Разширяемост на библиотеката чрез приставки (plugins);
* Между-браузърна поддръжка.

#### Toastr

За показването на различни съобщения на потребителя ще се използва javascript библиотеката Toastr. Въпреки че е много малка като размер, тя предлага много опции за персонализиране и настройка на съобщенията в желан формат. Някои от тях са – позиция, анимации, наличие на бутон за затваряне, колко дълго да бъде показано съобщението и др. Библиотеката предлага избор между 4 вида съобщения – за грешка, предупреждение, успех и информативни.

#### Карта

Основното преимущество на Trip Planner ще бъде неговата карта.

Двата най-популярни варианта за добавяне на карта към уебсайт са API-то на Google Maps и библиотеката Leaflet. И двата варианта са добри, надеждни и се припокриват в някои от функциите, които предоставят. За избор между двете трябва да се вземат предвид следните неща:

* Възможности:

Leaflet е създадена от самото начало, за да бъде лека и бърза. Много от функциите са умишлено изключени от основния код и са достъпни като приставки (plugins). Разработчикът сам решава кои функции иска да включи. Това може да изисква повече усилия, но от друга страна се получава точно това, което се иска, без излишни неща.

API-то на Google Maps, от друга страна, обединява всичко в един пакет. Това може да е удобно, тъй като не е нужно да се решава кои приставки (plugin) да се зареждат, но от друга страна в много случаи е излишно.

* Функции:

Тъй като и двете библиотеки са написани на JavaScript, вероятно всичко, което може да се направи в едната, може да се постигне и в другата, то просто може да изисква различно количество персонализиран код и /или приставки (plugin). По-долу е даден частичен списък от най-популярните функции, от които може да се интересуват много разработчици.

|  |  |  |
| --- | --- | --- |
| Функция | Leaflet | Google Maps API |
| Преместване/ приближаване | Вградено | Вградено |
| Мобилна поддръжка | Вградено | Вградено |
| Показване на векторни слоеве | Вградено | Вградено |
| Показване на изображения като слоеве | Вградено | Вградено |
| Клъстериране (групиране на точки) | С приставка (plugin) | С приставка (plugin) |
| Геокодиране | Няколко приставки налични | Вградено |
| Определяне на маршрут (routing) | Няколко приставки налични | Вградено |
| Street View | Може да се използва Google Street View | Вградено |
| Зареждане на данни от csv файл | С приставка (plugin) | С допълнителен код |
| Редактиране | С приставка (plugin) | С допълнителен код |

* Лицензи

Google Maps API-то е създадено и поддържано от Google. Всички актуализации, подобрения и правила относно използването му идват от Google. Текущата безплатна версия позволява до 25 000 зареждания на картатата и до 2500 заявки за геокодиране на ден. Всички безплатни внедрявания трябва да бъдат обществено достъпни, в противен случай трябва да се закупи премиум версия.

За разлика от това, Leaflet е проект с отворен код и няма почти никакви ограничения за използването му.

* Надграждания и версии

Тъй като Google контролира приложния програмен интерфейс (API) на Google Maps, те определят графика за пускане на нови функции. Те също така определят кога по-старите версии на кода ще бъдат оттеглени. Това означава, че повечето уебсайтове ще използват една и съща версия и че разработчиците ще знаят точно кои опции са налице.

При Leaflet подобренията са по-чести, нови версии се правят, тъй като се разработват нови функционалности. Постоянно се правят отделни актуализации и на приставките (plugin) към библиотеката. Все пак надграждането с нови актуализации макар препоръчително, не е задължително. Библиотеката може да се свали локално и тази нейна версия да се използва докато разработчикът желае това.

* Поддръжка

Ако се използва безплатната версия на Google Maps, може да се използва сайта им за поддръжка на общността. За платена версия, има на разположение други опции за поддръжка, включително гореща линия, на която можете да се позвъни.

За Leaflet много полезна информация, отговори на въпроси и решения на проблеми може да се потърси в StackExchange, StackOverflow и GitHub, където се съхранява кода на проекта.

И двата варианта са много добър избор и в повечето случаи това, което може да се постигне с единия, може и с другия. Затова много често изборът се свежда до лични предпочитания.

Ако вече се използват други услуги на Google, може да има смисъл да се избере Google Maps API-то . Ако разработчиците предпочитат приложния програмен интерфейс (API) на Google Maps или ако наистина харесват базовите карти на Google, всичкo това биха били основателни причини за избор.

За разработването на Trip Planner ще се използва Leaflet, тъй като документацията и приложния програмен интерфейс (API) са по-лесни за разбиранe. А широкото разнообразие от налични приставки улеснява добавянето на функционалност и е лесно да се създават персонализирани приставки, когато е необходимо.

Следва обобщение на нейните основни предимства:

* Безплатна и с отворен код;
* Малка и бърза;
* Лесен за разбиране и използване програмен интерфейс (API);
* Поддържа мобилни устройства;
* Много примери с много добра документация;
* Голямо разнообразие от приставки (plugin);
* Много добър първоначален дизайн за елементите на картата – маркери, контроли и др.;

#### Геокодиране (Geocoding)

Геокодирането е процес на преобразуване на адреси в географски координати (като географска ширина 37.423021 и дължина -122.083739), които могат да се използват за позициониране на карта или поставяне на маркери върху нея.

Обратното геокодиране е процесът на преобразуване на географските координати в адрес, четлив за хора.

Тъй като Trip Planner ще предоставя опция за търсене на точки по въведен адрес или град, трябва да се разработи геокодиране. Основният код на библиотеката Leaflet не предоставя такива функции, затова ще се използва приставка (plugin) - Leaflet Control Geocoder.

Leaflet Control Geocoder представлява прост геокодер за Leaflet, който по подразбиране използва OSM (Open Street Map) и Nominatum.

Open Street Map (OSM) е колаборативен проект за създаване на [свободна](https://bg.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE_%D1%81%D1%8A%D0%B4%D1%8A%D1%80%D0%B6%D0%B0%D0%BD%D0%B8%D0%B5) и редактируема [карта](https://bg.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D1%82%D0%B0) на света.

Проектът е стартиран от Стив Кост (Steve Coast) през [2004](https://bg.wikipedia.org/wiki/2004) г., с идеята за създаване на свободни за ползване карти на света, съставени и допълвани от доброволци, по подобие на [Уикипедия](https://bg.wikipedia.org/wiki/%D0%A3%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F). Проектът е алтернатива на различните несвободни подобни услуги на държавни и комерсиални организации като [Google Maps](https://bg.wikipedia.org/wiki/Google_Maps" \o "Google Maps) например. От тогава тя е нараснала до над 2 милиона регистрирани потребители, които могат да събират данни чрез ръчно проучване, GPS устройства, въздушна фотография и други безплатни източници. След това тези данни се публикуват под лиценза за отворена база данни.

През [2006](https://bg.wikipedia.org/wiki/2006) г., във Великобритания е регистрирана и фондация ОупънСтрийтМап (OpenStreetMap Foundation), която се занимава с поддръжка, организация, финансиране и лицензиране на проекта.

Nominatim е инструмент за търсене на OSM данни по име и адрес и за генериране на синтетични адреси на OSM точки (reverse geocoding).

За търсене на маршрути между две точки ще се използва OSRM (Open Source Routing Machine). OSRM е модерен маршрутизатор с отворен код за търсене на най-кратки маршрути в пътни мрежи. Тя е проектирана да работи добре с картографски данни от проекта OpenStreetMap. OSRM може да изчисли и изведе най-краткия път в рамките на няколко милисекунди. Тъй като е разработена с OpenStreetMap съвместимост, OSM файловете с данни могат лесно да бъдат импортирани.

# **Анализ**

## Концептуален модел

Trip Planner е уебсайт за запазване на любими географски локации и планиране на пътешествия, предназначен за използване на декстоп и мобилни устройства. Той е насочен към потребители, които искат удобно място за съхранение на любими места по света и лесен начин за филтриране на тези от тях, които са в близост до маршрута за следващата им екскурзия. Основните модули в него и техните функционалности са:

* Любими места – съдържа информация за всички добавени любими места, като местоположение (географски координати), информация, снимка, маркер и др.
* Карта – отговаря за инициализирането на географската карта, търсенето и изобразяването на локации върху нея, търсенето и изобразяването на маршрут, изобразяването на любимите места (всички или филтрирани в близост до точка или маршрут).
* Контроли – съдържа контроли за изчистване на всички елементи от картата, визуализиране на запазените маршрути, визуализиране на любимите места, смяна на максималната отдалеченост при филтрирането на близки места до маршрут, редактиране на любими места.
* База – съдържа функции за взаимодействие с базата като зареждане на всички любими места, добавяне на ново, редактиране, изтриване, зареждане на всички маршрути, добавяне на нов и изтриване.

## Функционални изисквания

Функционалните изисквания се отнасят до това какво системата трябва да може да прави. Те представляват твърдения за услугите, които системата трябва да доставя, как системата трябва да реагира на специфичен вход и как системата трябва да се държи в специфични ситуации.

Резултатът от анализирането и събирането на функционални изисквания е представяне на системата като функционална спецификация. Това е формален документ, който детайлизира функциите на системата. Обикновено този документ е съставен от отделни модули, организирани в йерархия. Това позволява описанието на системата да бъде разделено на податливи на обработка части, към които може да се подхожда индивидуално . Най-горното ниво на йерархията се състои от абстрактни описания на функциите на системата, докато на най-ниското ниво се определя в повече детйли какво трябва да представлява дадена функция.

Функционалните изисквания в Trip Planner се разделят на средните групи:

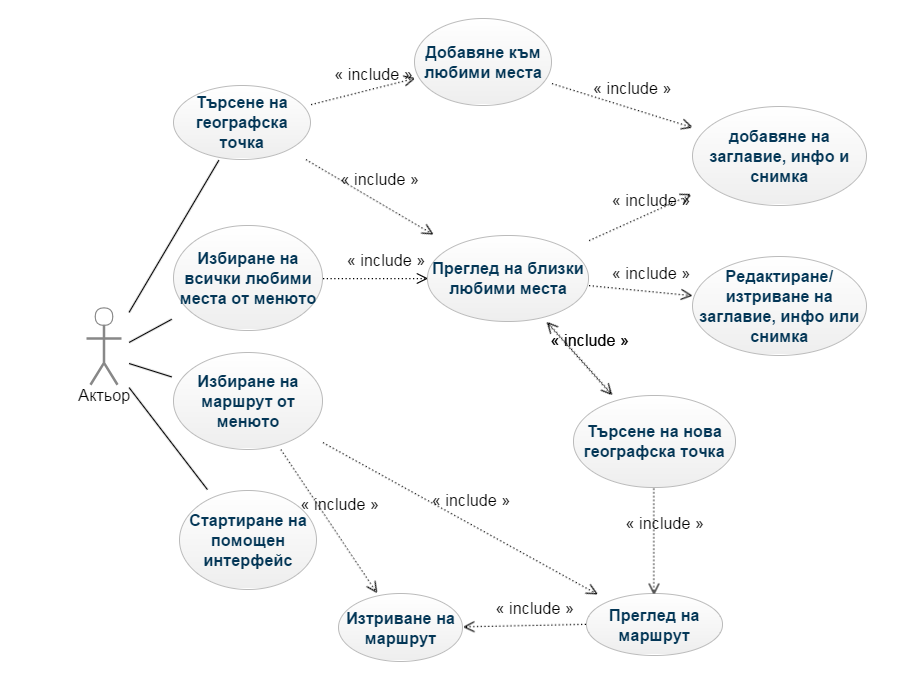
* Изисквания за любимите места:
  + Трябва да могат да се добавят нови места;
  + Всяко място задължително трябва да има локация върху географска карта;
  + За всяко място трябва да може да се добавя информация, заглавие и снимка;
  + При задаване на снимка за дадено място, тя трябва да стане и негов маркер върху картата;
  + При поместване на маркер на дадено място, трябва да се ъпдейтнат неговите координати;
  + Освен координатите трябва да могат да се редактират и заглавието, информацията и снимката на дадено място;
  + Трябва да има възможност за изтриване на места;
  + Трябва да има възможност за показване на всички места върху картата;
* Изисквания за построяването на маршрут:
  + Трябва да има възможност за търсене на точки върху картата по зададен адрес;
  + При въвеждането на всяка следваща точка, трябва да се изобразява маршрут между нея и предходно въведената;
  + Заедно с маршрута трябва да се изобразяват тези любими места, които са на разстояние по-малко от дадена константа;
  + Максималната отдалеченост на любимите места трябва да може да се задава от потребителя;
  + Всеки маршрут трябва да може да се запазва за повторно разглеждане;
  + Всеки маршрут трябва да може да се изтрива;
* Генерални изисквания:
  + След всяка извършена операция, трябва да се показва съобщение на потребителя – за успех, информативно или за грешка;
  + Трябва да се имплементира помощен интерфейс, който да води потребителя през основните функционалности стъпка по стъпка;

### Диаграма на потребителските случаи (Use case diagram)

В процеса на проектиране на софтуерния продукт, диаграмата на потребителските случаи е първата диаграма, която се създава от проектантите, когато се започне проект. Тази диаграма позволява да се опишат на най-високо ниво целите на потребителя, които системата трябва да изпълнява. Тези цели не е необходимо да са задачи или действия, а може да са по общи изисквания към функционалността на системата. С други думи това е техника за определяне на функционалните изисквания на една система. Те описват типичните взаимодействия между потребителите и системата, предоставят описание на начина, по който тя се използва.

Компоненти при Use case диаграми, използвани в Trip Planner:

* Actor (Актьор) – роля, която един потребител играе по отношение на системата. Може да бъде клиент, представител по поддръжка на клиенти, мениджър продажби, продуктов аналитик. Актьорите изпълняват случаите на употреба. Един актьор може да изпълнява много случаи на употреба и един случай на употреба може да се изпълнява от много актьори. Актьора клиент може да представлява множество хора. Един човек може да играе ролята на няколко актьора. Актьора може да не е човек, а друга компютърна система, ако системата извършва услуга за нея.
* Associations – плътна връзка без стрелка. Аssociations са връзка между actors и use cases и означава, че актьора осъществява потребителския случай.
* Includes – Връзка показваща, че един Use sace включва друг Use case. Примера илюстрира как и покупката с фактура и покупката онлайн се включват в сценария описан от покупателната оценка. В обобщение, includes връзката е да отстрани повторенията на сценария в съставните случаи. Може да се каже, че includes е като извикване на процедура.

диагр. 1. Диаграма на потребителските случаи в Тrip Planner

Примерите за използване (Use-cases) са основани на сценарий техника в UML, която идентифицира актьорите в едно взаимодействие и която описва самото взаимодействие. Множество от примери на използване би трябвало да опише всички взаимодействия със системата.

Use case е съставен от множество от сценарии. Сценариите са примери от реалния живот, как системата може да се използва.

Следват текстови описания на някои от потребителските случаи (сценарии).

* Добавяне на ново любимо място:

|  |  |
| --- | --- |
| Потребителски случай | Добавяне на ново любимо място |
| Главен актьор | Потребител |
| Цел | Да се добави ново любимо място |
| Условия | Актьорът е стартирал уебсайта. |
| Причина | Необходимост от добавяне на нови любими места |
| Главен сценарий | * Актьорът въвежда адрес в полето за търсене. На дадената локация се добавя маркер; * Актьорът кликва върху маркера. Отваря се прозорче с бутон – Add to my places; * Актьорът кликва върху бутона. Показва се съобщение за успешно добавено ново място и на същата позиция се показва нов маркер за мястото; |
| Алтернативни сценарии | Няма |
| Изключения | * Не е намерено място – върху картата не се променя нищо. Потребителят има възможност да търси друго място; * Вече има добавено място на дадената локация – ново място не се добавя и се показва съобщение за грешка; |

* Редактиране на снимката на любимо място:

|  |  |
| --- | --- |
| Потребителски случай | Редактиране на снимката на любимо място |
| Главен актьор | Потребител |
| Цел | Да се смени снимката на любимо място |
| Условия | * Актьорът е стартирал уебсайта. * Има визуализирано любимо място върху картата. |
| Причина | Да се смени иконата на маркера на любимото място |
| Главен сценарий | * Актьорът кликва върху маркера. Отваря се прозорче с името на мястото и бутон – More info; * Актьорът кликва върху бутона. Отваря се по-голям прозорец, съдържащ полета за име, информация и снимка; * Актьорът кликва върху иконата за редактиране в долния десен ъгъл на прозореца. Активира се възможността за редакция на полетата; * Актьорът намира подходяща снимка в интернет и копира нейния адрес; * Актьорът поставя адреса на снимката в полето – Marker icon. Снимката се визуализира незабавно в долната част на прозореца. При затваряне на прозореца се вижда, че иконата на маркера също е сменена; |
| Алтернативни сценарии | Няма |
| Изключения | Адресът на снимката е невалиден url адрес – остава предходната маркер икона; |

* Построяване на маршрут:

|  |  |
| --- | --- |
| Потребителски случай | Построяване на маршрут |
| Главен актьор | Потребител |
| Цел | Да се построи маршрут между две или повече точки |
| Условия | Актьорът е стартирал уебсайта. |
| Причина | Да се визуализират близките любими места |
| Главен сценарий | * Актьорът въвежда адрес в полето за търсене. На дадената локация се добавя маркер и се визуализират близките до нея места; * Актьорът въвежда втори адрес в полето за търсене. На втората локация също се добавя маркер. Също така се визуализира маршрут между двете точки и любимите места, които са отдалечени не повече от предварително въведено максимално разстояние; |
| Алтернативни сценарии | При всяко следващо въвеждане на адрес в полето за търсене, се построява маршрут между новата и предходно намерената локация. |
| Изключения | Не е намерено място – върху картата не се променя нищо. Потребителят има възможност да търси друго място; |

## Нефункционални изисквания

Нефункционални изисквания дефинират ограничения върху системата или процеса на разработка. Те произтичат от нуждите на клиента, от политиките на организацията, от нуждата от взаимодействие с вече съществуващи системи или от външни фактори като правилници и закони.

Нефункционални изисквания се групират в следните категории:

* изисквания към продукта - определят поведението на продукта (надеждност, производителност, ефективност и други);
* организационни изисквания - произтичат от правила и политики в организацията-поръчител и организацията-изпълнител;
* външни изисквания - изисквания за съвместимост, закони, наредби и други;

### Използваемост

Използваемостта се отнася до качеството на потребителското преживяване при взаимодействие с продукти или системи, включително уеб сайтове, софтуер, устройства или приложения. Използваемостта се характеризира с ефикасност, ефективност и цялостно удовлетворение на потребителя. Използваемостта е комбинация от фактори, включващи:

* Интуитивен дизайн: почти без усилие разбиране на архитектурата и навигацията на сайта
* Лекота на учене: колко бързо един потребител, който никога не е виждал потребителския интерфейс, може да изпълни основни задачи
* Ефективност на използването: Колко бързо опитен потребител може да изпълнява задачи
* Запомняне: след посещение на сайта, ако потребителят може да си спомни достатъчно, за да го използва ефективно при бъдещи посещения
* Честота и тежест на грешката: колко често потребителите правят грешки при използването на системата, колко сериозни са грешките и как се възстановяват потребителите от грешките
* Субективно удовлетворение: Ако потребителят обича да използва системата

Изискванията към използваемостта на уебсайт за добавяне на любими места и планиране на пътешествия са:

* системата трябва да е достатъчно проста и интуитивна, така че да не отнема много време за разучаване;
* добавянето на ново любимо място трябва да става максимално бързо и лесно;
* любимите места трябва да се изобразяват върху географска карта, като е обозначена точната им локация;
* за всяко любимо място трябва да могат да се добавят и редактират малко информация и снимка, но те трябва да се достъпват на друг екран, с цел да се избегне претрупване на информация върху картата;
* максималната отдалеченост на местата близки до даден маршрут трябва да може да се променя според желанията на потребителя;
* трябва да могат да се запазват построени маршрути и да се разглеждат след това;

### Изправност

Изправността се състои от три компонента:

* надеждност - устойчивостта към грешки (независещи от потребителя) на софтуерния продукт по време на работа;
* сигурност - степента на устойчивост и защита от вредни въздействия, които биха могли да навредят както на софтуерният продукт, така и на устройството;
* готовност - времето, през което софтуерният продукт е годен за използване;

Изискванията към изправността на уебсайт за запазване на места и планиране на пътешествия са:

* системата трябва да е достъпна за употреба по всяко време, за да пести време на потребителя;
* системата трябва да работи надеждно и при нормална употреба да няма грешки по време на използване на основните функции;
* системата трябва да е сигурна и използването ѝ да не довежда до неизправности в устройството;

### Модулност и изменяемост

Модулно програмиране е техника за проектиране на [софтуер](https://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80), увеличаваща степента, до която софтуерът е съставен от отделни, взаимозаменяеми компоненти, наречени модули. Това се постига чрез разбиване на функциите на програмата на модули, всеки от които изпълнява само една функция и съдържа всичко необходимо за нейното изпълнение. С други думи модулът е подсистема, част от [компютърна програма](https://bg.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8E%D1%82%D1%8A%D1%80%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0), която изпълнява конкретни действия. Понякога модулът се нарича още пакет, [библиотека](https://bg.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5)) и др. Модулът съдържа предимно по-малки единици (функции, класове и т.н.) в повечето случаи, написани на отделен [файл](https://bg.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB).

Литература:

<https://www.langoor.com/how-to-choose-the-right-technology-for-your-web-application/>

<https://www.apm.org.uk/blog/the-benefits-of-following-a-project-management-method/>

<http://www.seguetech.com/waterfall-vs-agile-methodology/>

<http://www.base36.com/2012/12/agile-waterfall-methodologies-a-side-by-side-comparison/>

<https://bg.wikipedia.org/wiki/Scrum>

<https://luminfire.com/2016/05/10/choosing-mapping-framework/>

<https://en.wikipedia.org/wiki/Open_Source_Routing_Machine>

<https://www.mongodb.com/document-databases>