Theories

1. SDLC:
   1. Що це таке?

**Жизненный цикл разработки ПО** (System/Software Development Life Cycle, SDLC) — процесс, состоящий из конкретных этапов, который начинается в момент принятия решения о необходимости создания ПО и заканчивается в момент прекращения поддержки ПО разработчиками.

Всего выделяют 7 основных этапов разработки [[1](https://ncube.com/blog/software-development-life-cycle-guide" \t "_blank)]:

1. Идея
2. Определение требований
3. Дизайн (архитектура) системы
4. Разработка
5. Тестирование
6. Развертывание
7. Поддержка
   1. Яка рзіниця між Waterfall та Scrum?

**Waterfall —** каскадная или водопадная модель. Традиционный подход. Представляет собой четко выделяемые последовательные стадии выполнения работ. Только после того, как закончены все работы на одной стадии, допускается переход к следующей.

**Agile —** гибкая методология разработки, состоящая из целого ряда подходов и практик, в основе который лежат быстрый результат и взаимодействие с людьми. К ним, в частности, относят такие методологии управления проектами, как Scrum и Kanban.

**Scrum —** подход, предполагающий решение задач в маленьких командах. Рабочий процесс при этом поделен на равные отрезки (спринты).

**Kanban —** уменьшение выполняющейся в данный момент работы (work in progress). Делает работу нагляднее за счет визуализации на доске. Допускает взятие срочных задач в разработку, не дожидаясь начала следующего этапа.

* Waterfall —Написание четкого плана и следование ему.
* Agile —Полная противоположность. Приветствуются изменения по ходу работы.
  1. Переваги та недоліки кожного з них

**Преимущества и недостатки метода Agile**

**К преимуществам метода относятся:**

* короткие итерации — циклы разработки длятся от 2 недели до 2 месяцев, по окончанию которых заказчик получает рабочую версию продукта
* высокая степень вовлечения исполнителей, организаторов и заказчиков проекта
* во главе угла стоит рабочий продукт как основной показатель прогресса — это можно рассматривать как плюс, так и минус, ведь в таком случае к команде проекта выдвигаются высокие требования по самоорганизации
* минимизация рисков благодаря гибкой системе внесения изменений;

**Не избежала методология и недостатков, которые органично «дополняют» её достоинства:**

* стимулирование постоянных изменений проекта: гибкость разработки продукта может привести к тому, что он никогда не дойдёт до финальной версии
* повышенные требования к квалификации и опыту команды: помимо непосредственно создания продукта команда должна анализировать возможные способы улучшения эффективности собственной работы, обмениваться информацией по проекту, быть мотивированной и самоорганизованной. Далеко не всегда ресурсы проекта позволяют привлечь таких специалистов
* сложность подсчёта итоговой суммы работы: стимуляция изменений и усовершенствования конечного продукта приводит к плавающему значению стоимости проекта. Поэтому Agile не подойдет для [управления проектами в строительстве](https://worksection.com/page/for-construction.html), где составляется четкая смета под всю работу.

**В число наибольших преимуществ методики Waterfall вошли:**

* понятная и простая структура процесса разработки — это снижает порог вхождения для команд
* удобная отчётность — можно легко отследить ресурсы, риски, затраченное время и финансы благодаря строгой этапности процесса разработки и детальной документации проекта
* стабильность задач — задачи, которые стоят перед продуктом, ясны команде с самого начала разработки, и остаются неизменными на протяжении всего процесса
* оценка стоимости и сроков сдачи проекта — сроки выпуска готового продукта, как и его итоговая стоимость могут быть просчитаны до момента запуска разработки.

**Среди недостатков водопадного метода можно выделить:**

* лишенный гибкости процесс — так, если проект требует больше временных и финансовых ресурсов, чем возможно, то под нож пойдёт фаза тестирования. Согласно исследованиям стоимость исправления багов после выпуска продукта выше в среднем в 20 раз, чем во время полноценного многоэтапного тестирования в процессе разработки
* «стойкость» к изменениям — жёсткий каркас из этапов разработки и условие предоставление только готового продукта определяют невозможность вносить изменения во время разработки
* инерционность — на первых стадиях прогноз временных и финансовых трат может измениться в сторону увеличения
* повышенный риск — классическая система тестирования подразумевает отдельно тестирование каждого из компонентов проекта, в том числе, во взаимодействии с другими. При использовании Waterfall происходит тестирование готового продукта.
  1. **Що таке SCRUM? Вміти пояснити, що з чого складається, й як прохожить процес**

**скрам (Scrum)** — это набор принципов, на которых строится процесс разработки, позволяющий в жёстко фиксированные и небольшие по времени итерации, называемые спринтами (sprints), предоставлять конечному пользователю работающее ПО с новыми возможностями, для которых определён наибольший приоритет.

Давайте для начала разберемся в основных ролях участников проекта, разрабатываемого по Scrum:

1. Product owner – это человек, который знает, как он хочет, чтобы выглядел продукт, т.е. он формирует список требований, которые заносятся в бэклог продукта идут в разработку
2. Team – это команда, которая непосредственно участвует в разработке. Очень важно чтобы она была самоорганизующейся и самоуправляемой, а также могла самостоятельно выделить какие из предложенных требований необходимо разработать в первую очередь.
3. Scrum Master – это человек, который следит за тем, чтобы соблюдались правила Scrum

Таким образом, идея есть, команда есть – можно начинать. Но для того, чтобы начать надо определиться, с планирования спринта. На этом этапе определяются функциональности, которые необходимо разработать и сколько времени на это потребуется. Далее начинается спринт.

Стоит отметить что неотъемлемой чертой скрама являются ежедневные митинги, в которых объясняется кто сколько чего сделал и остались/появились ли у него какие-то вопросы.

1. Software testing life cycle:
   1. Описати кожний етап: що на ньому роблять, які є артефакти тощо

STLC Model Phases

1. Requirement Analysis - test team studies the requirements from a testing point of view to identify testable requirements and the QA team may interact with various stakeholders (участниками) to understand requirements in detail
2. Test Planning - Senior QA manager determines the test plan strategy along with efforts and cost estimates for the project. Moreover, the resources, test environment, test limitations and the testing schedule are also determined.
3. Test case development -  involves the creation, verification and rework of test cases & test scripts after the test plan is ready.

* - Create test cases, automation scripts (if applicable)
* Review and baseline test cases and scripts
* Create test data (If Test Environment is available)

1. Test Environment setup - decides the software and hardware conditions under which a work product is tested. It is one of the critical aspects of the testing process and can be done in parallel with the Test Case Development Phase. Test team may not be involved in this activity if the development team provides the test environment. The test team is required to do a readiness check (smoke testing) of the given environment.
2. Test Execution -  in which testing of the software build is done based on test plans and test cases prepared. The process consists of test script execution, test script maintenance and bug reporting. If bugs are reported then it is reverted back to development team for correction and retesting will be performed.
3. Test Cycle closure - is completion of test execution which involves several activities like test completion reporting, collection of test completion matrices and test results. Testing team members meet, discuss and analyze testing artifacts to identify strategies that have to be implemented in future, taking lessons from current test cycle. The idea is to remove process bottlenecks for future test cycles.
4. [Software testing types](http://savesoftwaretesting.com/moodle/mod/resource/view.php?id=285):
   1. На які типи тестування поділяється згідно ISTQB?

**Depending on its objectives, there are four software testing types:**

* Functional testing.- verifies that each function of the software application operates in conformance with the requirement specification. Functional testing shows **“What the system does”.** The goal of this testing is to check whether the system is functionally perfect.

Non-functional testing - тестирование свойств, которые не относятся к функциональности системы. Данные свойства определяются нефункциональными требованиями, которые характеризуют продукт с таких сторон, как:

**Надежность** (реакция системы на непредвиденные ситуации).

**Производительность** (Работоспособность системы под разными нагрузками).

**Удобство** (Исследование удобности работы с приложением с точки зрения пользователя).

**Масштабируемость** (Требования к горизонтальному или вертикальному масштабированию приложения).

**Безопасность** (Защищенность пользовательских данных).

**Портируемость** (Переносимость приложения на различные платформы).

* Structural testing - checks the implementation of the program or code via testing of the structure of the software system or its components **тестирование, которое учитывает внутренние механизмы системы или компонента** (ISO/IEC/IEEE 24765). Обычно включает тестирование ветвей, маршрутов, операторов (см. покрытие кода). Statement Coverage verifies the each statement in a program is executed at least once during program testing.
* Change related testing. is provided to ensure that previously eradicated bugs have been fixed and to catch bugs that may have been accidentally appeared into a new version. According to these goals, there are two subtypes of Change related testing: **Confirmation testing (Re-testing)** and **Regression testing.**

Firstly you should perform **Confirmation testing** to ensure the bug has indeed been successfully removed.

**Regression testing** consists not only of the detected bug test cases. It is performed not only for verifying bug fixing but also to ensure that new defects have not come up or discovered after the changes.

* 1. Які є типи тестування? Описати кожен типу тестування

1. **Functional:**
   1. **Installation testing** - Installation testing, intended to check the successful installation and upgrade or remove the program.
   2. **Smoke Testin**g - Smoke testing is performed in order to show that the most necessary functionality of the program (which requires the product).

This testing is done by developers or testers.

* 1. **Functionality Testing** - Functionality testing is done to check that the software operates correctly in accordance with design specifications.

Installation and configuration on the local machine

Enter text, including using non-Latin characters or extended.

The main functions of the applications that have not been tested during Smoke testing.

Exact hotkey shortcuts without any duplication.

* 1. **Compatibility testing** - Checks whether the application or software compatible with the hardware, operating system, database or other software systems.

1. **Non-functional:**
   1. **Security testing** –

determining what information system protects data and maintains functionality as intended.

Security testing, in general, this type of test that checks whether a program or product is protected or not.

This is a test whether the system is vulnerable to attack if anyone can break the system or enter without any permission.

* 1. **Usability testing** - is necessary to verify that the user interface is easy to use and understand

Usability testing can be checked:

Time on Task – How long it takes people to complete basic tasks? (For example, to find something to buy, create a new account and order detail).

Accuracy – How many mistakes did people?

Recall – How people can recall specific action steps after some time did not use the system?

Emotional response – How people will feel themselves after finished task? (Are yon sure? Will users recommend this program to friends?)

* 1. **Localization testing and Internationalization testing -** Localization (L10N) testing checks how well the application under test has been Localized into a particular target language.

Internationalization (I18N) testing checks if *all data/time/number/currency formats* are displayed according to selected locale and if all language specific characters are displayed.

* 1. **Performance testing** - направленных на воссоздание пользовательских запросов в системе и сравнение ожидаемых результатов с полученными показателями, а также определение скорости процедур, стабильности, надежности и масштабируемости системы в целом. Полученные результаты позволяют обнаруживать уязвимости с пропускной способностью приложения, временем загрузки, обработкой больших объемов данных и предотвращением их использования в приложении.
  2. **Load testing** - под нагрузочным тестированием понимается практика моделирования ожидаемого использования приложения с помощью эмуляции работы нескольких пользователей одновременно. Таким образом, подобное тестирование больше всего подходит для многопользовательских систем, чаще — использующих клиент-серверную архитектуру (например, веб-серверов).имитирующее работу определенного количества бизнес пользователей на каком-либо общем (разделяемом ими) ресурсе.
  3. **Stress testing -** form of testing that is used to determine the stability of a given system or entity.

The idea is to stress a system to the breaking point in order to find bugs that will make that break potentially harmful.

* 1. **UI testing** - User Interface testing is done to verify that the application interface to defined standards.

1. **Confirmation testing** - confirms that the bug has been sent**,** It is important to ensure that the test is executed in exactly the same way it was the first time using the same inputs, data and environments.

**Regression testing -** Regression testing is to verify that the changes made in the software (if corrected old bugs) have not led to the emergence of new bugs.

| **параметр** | **Тестирование черного ящика** | **Тестирование белого ящика** |
| --- | --- | --- |
| **Определение** | Это метод тестирования, который используется для тестирования программного обеспечения без знания внутренней структуры программы или приложения. | Это подход к тестированию, при котором внутренняя структура известна тестеру. |
| **кличка** | Он также известен как управляемое данными, блочное тестирование, тестирование данных и функциональное тестирование. | Это также называется структурное тестирование, тестирование прозрачного бокса, тестирование на основе кода или тестирование прозрачного бокса |

* 1. Яка різниця між Box-testing?

|  |  |  |
| --- | --- | --- |
| **База тестирования** | Тестирование основано на внешних ожиданиях; внутреннее поведение приложения неизвестно. | Внутренняя работа известна, и тестер может тестировать соответственно. |
| **Применение** | Этот тип тестирования идеально подходит для более высоких уровней тестирования, таких как тестирование системы, приемочное тестирование. | Тестирование лучше всего подходит для более низкого уровня тестирования, такого как модульное тестирование, интеграционное тестирование. |
| **Знание программирования** | Знания по программированию не нужны для тестирования Black Box. | Знания по программированию необходимы для проведения тестирования в Белом Ящике. |
| **Внедрение знаний** | Знания о внедрении не требуют тестирования Black Box. | Полное понимание требует реализации тестирования WhiteBox. |
| **автоматизация** | Тест и программист зависят друг от друга, поэтому сложно автоматизировать. | Тестирование белого ящика легко автоматизировать. |
| **Задача** | Основная цель этого тестирования – проверить, какая функциональность тестируемой системы. | Основная цель тестирования White Box делается для проверки качества кода. |
| **Проверено** | Выполняется конечным пользователем, разработчиком и тестером. | Обычно делается тестером и разработчиками. |
| **Метод тестирования** | Он основан на методе проб и ошибок. | Область данных и внутренние границы могут быть проверены. |
| **Время** | Это менее исчерпывающий и трудоемкий процесс. | Исчерпывающий и трудоемкий метод. |
| **Алгоритм теста** | Не лучший метод для тестирования алгоритмов. | Лучше всего подходит для тестирования алгоритмов. |
| **Доступ к коду** | Доступ к коду не требуется для тестирования черного ящика. | Тестирование белого ящика требует доступа к коду. Таким образом, код может быть украден, если тестирование выполняется на стороне. |
| **Выгода** | Хорошо подходит и эффективен для больших сегментов кода. | Это позволяет удалить лишние строки кода, которые могут привести к скрытым дефектам. |
| **Уровень мастерства** | Низкоквалифицированные тестировщики могут тестировать приложение, не зная о реализации языка программирования или операционной системы. | Требуется опытный тестировщик с огромным опытом для тестирования белого ящика. |
| **Недостатки** | Обновление сценария тестирования автоматизации необходимо, если вы часто изменяете приложение. | Автоматизированные тестовые случаи могут стать бесполезными, если кодовая база быстро меняется. |

* 1. Яка різниця між Verification та Validation? - **валидация подтверждает, что «вы создали правильный продукт» (который устраивает клиента), а верификация подтверждает, что «вы создали продукт таким, каким видите его и намеревались его сделать»**.
  2. Яка різниця між Proactive та Reactive testing - Proactive - An approach (подход) in which (в котором) the test design process (процесс тест дизайна) is initiated (начинается) as early as possible in order to find and fix the defects before the build is created. Reactive - An approach in which the testing is not started until after design and coding are completed.
  3. Які переваги та недоліки Manual та Automated testing –

**Advantages of Automated Testing**

* Automation testing is faster in execution
* It is cheaper compared to manual testing in a long run
* Automated testing is more reliable
* Automated testing is more powerful and versatile (универсальный)
* It is mostly used for regression testing
* It is reusable because automation process can be recorded
* It does not require human intervention. Test scripts can be run unattended
* It helps to increase the test coverage

**Disadvantages of Automated Testing**

* It is recommended only for stable products
* Automation testing is expensive initially
* It has some limitations such as handling captcha, getting visual aspects of UI such as fonts, color, sizes etc.,

**Advantages of Manual Testing**

* Manual testing can be done on all kinds of applications
* It is preferable for short life cycle products
* Newly designed test cases should be executed manually
* Application must be tested manually before it is automated
* It is preferred in the projects where the requirements change frequently
* It is cheaper in terms of initial investment compared to Automation testing
* It requires less time and expense to begin productive manual testing
* It allows tester to perform adhoc testing
* There is no necessity to the tester to have knowledge on Automation Tools

**Disadvantages of Manual Testing**

* Manual Testing is time-consuming mainly while doing regression testing.
* Manual testing is less reliable compared to automation testing because it is conducted by humans. So there will always be prone to errors and mistakes.
* Expensive over automation testing in the long run

It is not possible to reuse because this process can’t be recorded

1. Яка різниця між Confirmation та Regression testing? - **Confirmation testing** - confirms that the bug has been sent**,** It is important to ensure that the test is executed in exactly the same way it was the first time using the same inputs, data and environments.

**Regression testing -** Regression testing is to verify that the changes made in the software (if corrected old bugs) have not led to the emergence of new bugs.

* 1. Коли виконуємо регрешн тестування? - Регрессионное тестирование выполняется **при внесении изменений в существующие функциональные возможности программного обеспечения или, если есть ошибка исправления в программном обеспечении**.
  2. Що таке Scripted та Unscprited testing?- Scripted Testing is a type of [software testing](https://www.geeksforgeeks.org/software-testing-basics/) that is performed by organizing the details of all the tasks and planning. Testing team properly plans for the scripted testing and a test script is written.

Unscripted Testing: **Unscripted Testing is a type of software testing in which the tester is free to select any possible methodology to test the software.** In unscripted testing, software developers use their personal learning, knowledge, skills and abilities to test the software developed by themselves.

1. Test design technique:
   1. Різновиди та де їх використовувати

**Статические и динамические**. Статические призваны предотвратить ошибки еще до самого тестирования продукта (не требует прямого использования продукта).

Informal review

This is one of the type of review which doesn't follow any process to find errors in the document. Under this technique, you just review the document and give informal comments on it.

**Walkthrough:**The author of the work product explains the product to his team. Participants can ask questions if any.  Meeting is led by the author*.* QC vs QA

Technical review

This review concentrates mainly on the technical document related to the software such as **Test Strategy, Test Plan** and requirement specification documents.

Inspection

The main purpose is to find defects and meeting is led by trained moderator.

This review is a formal type of review where it follows strict process to find the defects.

Reviewers have checklist to review the work products .They record the defect and inform the participants to rectify those errors.

**Динамическое тестирование**

#### **Техники тест дизайна:**

* **Эквивалентное Разделение (Equivalence Partitioning — EP)**. Как пример, у Вас есть диапазон допустимых значений от 1 до 10, Вы должны выбрать одно верное значение внутри интервала, скажем, 5 и одно неверное значение вне интервала — 0.
* **Анализ Граничных Значений (Boundary Value Analysis — BVA).** Если взять пример выше, в качестве значений для позитивного тестирования выберем минимальную и максимальную границы (1 и 10) и значения больше и меньше границ (0 и 11). Анализ Граничных значений может быть применен к полям, записям, файлам, или к любого рода сущностям, имеющим ограничения.
* **Decision table** Customer can take a loan in a bank.
* The bank gives loan application, where customer can enter the amount of the monthly re-payment or the number of years customer wants to take to pay it back (the term of the loan).
* Customer should infill only one of the proposed fields.
* If customer enter both, then customer will get an error message.
* Also customer can choose a discount which helps reduce %.
* **State transition.** Customer chooses Arabica coffee from coffee machine.
* He selects specific sort of coffee (in this case Arabica), and enters money.
* If Arabica coffee is available, then machine will check money.
* If Arabica coffee is not available, then customer will get proper message and his money back.
* If not enough money is entered, then machine will ask to enter more.
* If amount of money is OK, then customer will get coffee and his change in a minute.

**Диаграмма пользовательских ролей» (Use Case Diagram).**

Представим себе обычный интернет-магазин. Какие там есть роли? Например:

* Администратор
* Продавец
* Покупатель (зарегистрированный)
* Незарегистрированный пользователь

У каждой роли есть свои права и доступы, и каждый из них умеет что-то делать. Поэтому рисуем четырех человечков, подписываем их роли и переходим к их действиям. Все действия будем указывать в небольших прямоугольниках. Проведем стрелочку от роли к доступному действию.

**Парное тестирование (Pairwise Testing — PT)** Для техники «Парное тестирование» нужно открыть любой интернет-магазин и каталог товаров. Пусть это будут «Cмартфоны, ТВ и электроника».

Давайте посмотрим на левую панель. Здесь есть обширный фильтр: по продавцу, отправке, бренду, алфавиту, цене, беспроцентному кредиту и т.д. В чем суть парного тестирования? Мы имеем много разных характеристик, по которым нужно сортировать, но мы будем тестировать не по одной характеристике, а сразу по двум. Для чего? Чтобы проверить, какая будет реакция у системы, какой будет результат. Переходим к практике. Сначала мы должны протестировать продавца и готовность к отправке. Ставим две галочки:

[**Угадывание ошибок (Error Guessing)**](https://highload.today/blogs/8-tehnik-test-dizajna-s-primerami/#7)

* Уменьшать ширину окна по чуть-чуть, чтоб отловить баги в User Interface
* Делить на ноль
* Ввести пустое значение в поле
* Ввести пробел в начало, в середину, в конец поля
* Негативное тестирование дат (30, 31 февраля, а также даты, которые еще не наступили)
* Ставить на аватарку текстовые файлы, музыку, пустые файлы

1. Principles of testing: скільки їх та що вони означають –

Существует **7 принципов тестирования**:

1. Тестирование демонстрирует наличие [дефектов](https://beqa.pro/blog/%d0%b1%d0%b0%d0%b3-%d0%b8-%d0%b1%d0%b0%d0%b3-%d1%80%d0%b5%d0%bf%d0%be%d1%80%d1%82/), а не их отсутствие - Тестирование снижает вероятность наличия дефектов, находящихся в программном обеспечении, но, даже если дефекты не были обнаружены, **тестирование не доказывает корректность работы ПО**.
2. Исчерпывающее тестирование недостижимо – **Полное тестирование** с использованием всех комбинаций вводов и предусловий физически **невыполнимо**(за исключением тривиальных случаев). Вместо попытки исчерпывающего тестирования должны использоваться анализ рисков, методы тестирования и расстановка приоритетов, что бы сосредоточить усилия по тестированию.
3. Раннее тестирование сохраняет время и деньги - Для нахождения дефектов на ранних стадиях разработки, [статические](https://beqa.pro/blog/testing-types/#white-box-testing) и [динамические](https://beqa.pro/blog/testing-types/#black-box-testing) активности по тестированию должны быть **начаты как можно раньше** в [жизненном цикле разработки программного обеспечения](https://beqa.pro/blog/sdlc/). Тестирование на ранних этапах жизненного цикла разработки программного обеспечения помогает сократить или исключить дорогостоящие изменения.
4. Кластеризация дефектов - **принцип, который предполагает, что небольшое количество модулей содержат в себе большинство багов**. Это яркий пример применения в тестировании принципа Парето: 80% проблем таятся в 20% модулей. Имея опыт, можно легко определить эти проблемные модули.
5. Парадокс пестицида - Если есть некий набор повторяемых тестов, он будет бесполезен в определении новых дефектов. Чтобы решить эту проблему, тест-кейсы должны регулярно просматриваться и обновляться, должны добавляться новые тест-кейсы, ориентированные на другую функциональность или тестирующие уже существующую под другим углом.
6. Тестирование зависит от контекста- способ, которым вы тестируете сайт для e-commerce, будет отличаться от способа тестирования мобильного приложения. Софт бывает самый разный и подход к его тестированию тоже бывает самый разный.
7. Заблуждение об отсутствии ошибок - Иногда тестрируя и выискивая функциональные баги, мы забываем посмотреть с другой стороны и спросить а нужно ли это пользователю. Если эта фитча не соответствует ожиданиям пользователя и его потребностям, то какой бы качественный наш продукт не был — это уже не так важно.
8. Traceability and Dependency matrices

**Traceability Matrix**

Это двумерная таблица, содержащая соответствие функциональных требований (**functional requirements**) продукта и подготовленных тестовых сценариев (**test cases**). В заголовках колонок таблицы расположены требования, а в заголовках строк — тестовые сценарии. На пересечении — отметка, означающая, что требование текущей колонки покрыто тестовым сценарием текущей строки. Матрица обычно хранится в виде электронной таблицы.

**Матрица соответствия требований** используется **QA**-инженерами для валидации покрытия требований по продукту тестами. Цель «Traceability Matrix» состоит в том, чтобы выяснить:

* какие требования «покрыты» тестами, а какие нет.
* избыточность тестов (одно функциональное требование покрыто большим количеством тестов).

1. [Requirements](http://savesoftwaretesting.com/moodle/mod/resource/view.php?id=302):
   1. Види –

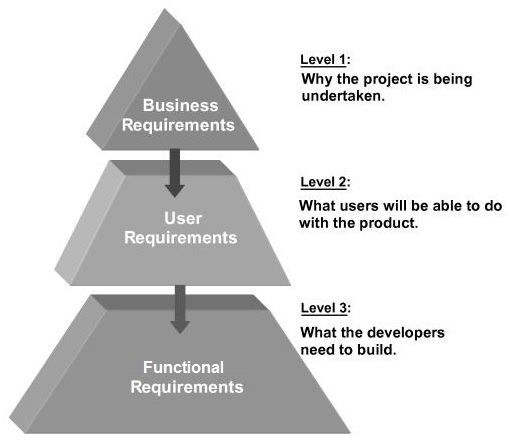
**Functional:**

Business requirements

User requirements

Functional requirements

* **Nonfunctional: требования, определяющие свойства, которые система должна демонстрировать, или ограничения, которые она должна соблюдать, не относящиеся к поведению системы**. Например, производительность, удобство сопровождения, расширяемость, надежность, факторы эксплуатации.
  + Physical environment (где будет использоваться продукт)
  + Performance (как будет исполнятся продукт)
  + Documentation (документация к продукту)
  + Quality attributes (атрибуты качества к продукту)
  1. Типи вимоги



* 1. **Хороші характеристики вимог**

**Necessary - Does the requirement present some business need**

**Complete - On loss of power, the battery backup must support operations for 20 minutes.**

**Consistent (последовательный)**

**Unambiguous (однозначный, недвусмысленный) - : When the user accesses any screen, it must appear on the monitor within 2 seconds**

**Verifiable (проверяемость) The user interface shall be menu driven. It shall provide dialog boxes, help screens, radio buttons, dropdown list boxes, and spin buttons for user inputs.**

1. Client server architecture - overview:

**Описати, як воно працює** - сборное понятие, состоящее из двух взаимодополняющих компонентов: сервера и, собственно, клиента.

**Клиент** – локальный компьютер на стороне виртуального пользователя, который выполняет отправку запроса к серверу для возможности предоставления данных или выполнения определенной группы системных действий.

**Сервер** – очень мощный компьютер или специальное системное оборудование, которое предназначается для разрешения определенного круга задач по процессу выполнения программных кодов. Он выполняет работы сервисного обслуживания по клиентским запросам, предоставляет пользователям доступ к определенным системным ресурсам, сохраняет данные или БД.

Особенности такой модели заключаются в том, что пользователь отправляет определенный запрос на сервер, где тот системно обрабатывается и конечный результат отсылается клиенту. В возможности сервера входит одновременное обслуживание сразу нескольких клиентов.

Если одновременно поступает более одного запроса, то такие запросы устанавливаются в определенную очередь и сервером выполняются по очереди. Порой запросы могут иметь свои собственные приоритеты.

**Які протокли використовуються? - TCP/IP** – совокупность протоколов передачи информации. TCP/IP – это особое обозначение всей сети, которая функционирует на основе протоколов TCP, а также IP.

**TCP** – вид протокола, который является связующим звеном для установки качественного соединения между 2 устройствами, передачи данных и верификации их получения.

**IP** – протокол, в функции которого входит корректность доставки сообщений по выбранному адресу. При этом информация делится на пакеты, которые могут поставляться по-разному.

**MAC** – вид протокола, на основании которого происходит процесс верификации сетевых устройств. Все устройства, которые подключены к сети Интернет, содержат свой оригинальный MAC-адрес.

**ICMP** – протокол, который ответственен за обмен данными, но не используется для процесса передачи информации.

**UDP** – протокол, управляющий передачей данных, но данные не проходят верификацию при получении. Этот протокол функционирует быстрее, чем протокол TCP.

**HTTP** – протокол для передачи информации (гипертекста), на базе которого функционируют все сегодняшние сайты. В его возможности входит процесс запрашивания необходимых данных у виртуально удаленной системы (файлы, веб-страницы и прочее).

**FTP** – протокол передачи информации из особого файлового сервера на ПК конечного пользователя.

**POP3** – классический протокол простого почтового соединения, который ответственен за передачу почты.

**SMTP** – вид протокола, который может устанавливать правила для передачи виртуальной почты. Он ответственен за передачу и верификацию доставки, а также оповещения о возможных ошибках.

* 1. **Що таке порт** это число, которое идентифицирует назначение сетевых потоков данных в пределах одного **компьютера**
  2. Де відбувається шифрування, якщо використовується протокол HTTPS? – шифрование на сервере, а расшифровка у клиента

1. TCP/IP stack - <https://ukr.kagutech.com/3980056-tcp-ip-protocol-stack-structure-levels-configuration-internet-protocols> - Он используется в качестве основы для глобальной сети интернет. сетевая модель, описывающая процесс передачи цифровых данных. Она названа по двум главным протоколам, по этой модели построена глобальная сеть — интернет
2. OWASP 10 - <https://owasp.org/www-project-top-ten/> - Использование рекомендаций Open Web Application Security Project (OWASP) давно стало стандартом обеспечения безопасности веб-приложений. это отчет или информационный документ, в котором перечислены основные проблемы, связанные с безопасностью веб-приложений. Он регулярно обновляется, чтобы постоянно отображать 10 наиболее серьезных рисков, с которыми сталкиваются организации. OWASP рекомендует всем компаниям учитывать выводы документа при построении корпоративных процессов, чтобы минимизировать и смягчить актуальные риски безопасности.
3. REST API:
   1. **Поняття, яка різниці між REST та SOAP -** REST — это архитектурный стиль. SOAP — это формат обмена сообщениями

   
Пример реализации **RESTful**: JSON через HTTP

 Пример реализации **SOAP**: XML поверх SOAP через HTTP

Специфика SOAP — это формат обмена данными. С SOAP это всегда SOAP-XML, который представляет собой XML,

#### **Формат обмена сообщениями**

* В SOAP вы используете формат SOAP XML для запросов и ответов.
* В REST такого фиксированного формата нет. Вы можете обмениваться сообщениями на основе XML, JSON или любого другого удобного формата. JSON является самым популярным среди используемых форматов.

#### **Определения услуг**

* SOAP использует **WSDL** (Web Services Description Language) — язык описания веб-сервисов и доступа к ним, основанный на языке XML.
* REST не имеет стандартного языка определения сервиса. Несмотря на то, что WADL был одним из первых предложенных стандартов, он не очень популярен. Более популярно использование Swagger или Open API.
  1. Методи - GET используется для получения (или чтения) представления ресурса

Метод PUT обычно используется для предоставления возможности обновления ресурса. Тело запроса при отправке PUT-запроса к существующему ресурсу URI должно содержать обновленные данные оригинального ресурса (полностью, или только обновляемую часть).

PATCH запрос используется для \*\*модификации\*\* ресурса. PATCH запрос должен содержать только изменяемые данные ресурса, а не все его данные.

POST запрос наиболее часто используется для создания новых ресурсов.

DELETE запрос крайне прост для понимания. Он используется для удаления ресурса, идентифицированного конкретным URI (ID).

Разница patch put PATCH используется для частичного изменения ресурса. PUT создает новый ресурс или заменяет представление целевого ресурса

* 1. Які будуть HTTP відповіді відповідно до практичних задач

GET - HTTP 200 (OK). В случае наличия ошибок обычно возвращается код 404 (NOT FOUND) или 400 (BAD REQUEST).

PUT - При успешном обновлении посредством выполнения PUT запроса возвращается код 200 (или 204 если не был передан какой-либо контент в теле ответа). Если PUT используется для создания экземпляра — обычно возвращают HTTP код 201 при успешном создании.

DELETE При успешном удалении возвращается 200 (OK) код HTTP, совместно с телом ответа, содержащим данные удалённого ресурса. Также возможно использование HTTP кода 204 (NO CONTENT) без тела ответа.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 201 | Created |  | |  | | --- | |  | |
| 401 | Unauthorized |  | |  | | --- | |  | |
| 403 | Forbidden |  | |  | | --- | |  | |
| 404 | Not Found |  |  |
| 204 | No Content | | |

1. MySQL:
   1. **Що таке реляційна база даних? -** Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая стока таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности. Каждая строка в таблице может быть помечена уникальным идентификатором, называемым первичным ключом, а строки из нескольких таблиц могут быть связаны с помощью внешних ключей. К этим данным можно получить доступ многими способами, и при этом реорганизовывать таблицы БД не требуется.
   2. Що таке primary key та foreign key? - **PRIMARY KEY** - первичный ключ таблицы. UNIQUE - уникальный ключ таблицы. **FOREIGN KEY** - внешний ключ, обеспечивает ссылку на другую таблицу и гарантирует ссылочную целостность между родительской и дочерней таблицами.
   3. Яка різниця мід SQL та NoSQL? - SQL и NoSQL, реляционные и нереляционные базы данных

**реляционные БД** используют для определения и обработки данных SQL — гибкий язык запросов, выбор которого минимизирует ряд рисков и прекрасно подходит для комплексных запросов. Но есть и обратная сторона медали: SQL имеет ряд ограничений. В частности, построение запросов обязывает предопределять структуру данных, то есть, если вы потом захотите эту структуру изменить, это может оказаться губительным для системы

**нереляционных БД**, то они предлагают динамическую структуру данных, то есть данные можно хранить разными способами: [колоночным](https://otus.ru/nest/post/1642/), [документо-ориентированным](https://otus.ru/nest/post/1619/), на основе [пар «ключ-значение»](https://otus.ru/nest/post/1603/) либо в виде [графов](https://otus.ru/nest/post/2016/).

Что на практике может дать такая гибкость: • возможность создавать документы без необходимости заранее задавать их структуру; • каждый документ способен обладать своей структурой; • у каждой БД возможно наличие собственного синтаксиса; • есть возможность добавлять поля непосредственно в процессе работы с данными.

* 1. Приклади запитів –

## SELECT

## WHERE

## GROUP BY

## HAVING

## ORDER BY

## BETWEEN

## LIKE

Оператор LIKE используется в WHERE, чтобы задать шаблон поиска похожего значения.

## IN

С помощью IN можно указать несколько значений для оператора WHERE:

## JOIN

JOIN используется для связи двух или более таблиц с помощью общих атрибутов внутри них.

1. HTML/CSS – разметка страницы, используется для написания веб сайтов. Css отвечает за внешний вид ХТМЛ, работают в связке
2. Authentication-vs-Authorization - <https://www.okta.com/identity-101/authentication-vs-authorization/>

**Аутентификация** используется для подтверждения личности зарегистрированного пользователя. Проверка подлинности – это процесс проверки учетных данных: идентификатора пользователя (имени, адреса электронной почты, номера телефона) и пароля.

**Авторизация**

Происходит после того, как личность пользователя успешно аутентифицируется системой. Процесс авторизации определяет, имеет ли прошедший проверку человек доступ к определенным ресурсам: информации, файлам, базе данных.

1. File Systems​ - depends on operation system - <http://www.ufsexplorer.com/und_fs.php>

Microsoft Windows employs two major file systems: **NTFS** New Technology File System, the primary format most modern versions of this OS use by default, and **FAT** File Allocation Table, which was inherited from old DOS and has **exFAT** as its later extension

ReFS (Resilient File System) is the latest development of Microsoft introduced with Windows 8 and now available for Windows 10. Its architecture absolutely differs from other Windows formats and is mainly organized

HPFS (High Performance File System) was created by Microsoft in cooperation with IBM and introduced with OS/2 1.20 in 1989 as a file system for servers that could provide much better performance when compared to FAT

## File systems of macOS

HFS+ used to be the primary format of Apple desktop products, including Mac computers, iPods, as well as Apple X Server products before it was replaced by APFS in macOS High Sierra. Volumes are divided into sectors, typically 512 bytes in size

### APFS

The Apple file system is aimed to address fundamental issues present in its predecessor and was developed to efficiently work with modern flash storages and solid-state drives. This 64-bit format uses the copy-on-write method to increase performance, which allows to copy each block before the changes to it are applied, and offers a lot of data integrity and space-saving features.

## File systems of Linux

### Ext

Ext2, Ext3, Ext4 are simply different versions of the "native" Linux Ext file system. This type falls under active developments and improvements.

### ReiserFS

ReiserFS - an alternative Linux file system optimized for storing a huge number of small files.

### XFS

XFS - a robust journaling file system that was initially created by Silicon Graphics and used by the company's IRIX servers. In 2001, it made its way to the Linux kernel and is now supported by most Linux distributions, some of which, like Red Hat Enterprise Linux, even use it by default. This FS type is optimized for storing very big files and volumes on a single host.

### JFS

JFS - a file system developed by IBM for the company's powerful computing systems. JFS1 usually stands for **JFS**, **JFS2** is the second release. Currently, this project is open-source and implemented in most modern Linux versions.

F2FS – a Linux file system designed by Samsung Electronics that is adapted to the specifics of storage devices based on the NAND flash memory that are widely used in modern smartphones and other computing systems.

### Btrfs

Btrfs - a file system based on the copy-on-write principle (COW) that was designed by Oracle and has been supported by the mainline Linux kernel since 2009. Btrfs embraces the features of a **logical volume manager,**being able to span multiple devices, and offers much higher fault tolerance, better scalability, easier administration, etc. together with a number of advanced possibilities.

## File systems of BSD, Solaris, Unix

The most common file system for these operating systems is UFS (Unix File System) also often referred to as FFS (Fast File System).

1. Types of mobile applications:  
   1. **Browser-based -** Данный тип приложений использует технологию WEB для возможности работы на мобильном устройстве. Главным и несомненным преимуществом таких приложений является возможность единоразового создания на все типы платформ. Также мобильная версия сайта имеет весомое преимущество – кроссплатформенность. Однако из минусов стоит отметить, что такие приложения не позволяют использовать функции камеры или геолокации в смартфоне, а также их невозможно загрузить из магазина приложений.
   2. **Native Device Application** - Они создаются на основе языков программирования для таких операционных систем, как Android, Windows Phone и iOS. Благодаря таким приложениям можно в полной мере использовать функционал GPS, видеокамер или датчиков ускорения. Главным преимуществом нативных приложений является возможность автономной работы без необходимости подключения к интернету. Чаще всего приложения такого типа распространяются через магазины приложений Play Market, AppStore и другие. Также нативными приложениями могут являться различные электронные библиотеки.​
   3. **Hybrid Applications -** это соединенные воедино нативные и веб-приложения. Главными преимуществами гибридных приложений являются кроссплатформенность на web-технологиях и возможность доступа к функциям смартфона или другого мобильного устройства.