***Протокол встречи с заказчиком***

1. ПО Должно носить название “NUOS Library”.

2. ПО должно поддерживать размещение методичек.

3. ПО должно поддерживать авторизацию пользователя.

4. ПО должно содержать вкладку с последними просмотренными документами.

5. ПО должно содержать вкладку с сохраненными документами.

6. ПО должно содержать вкладку с профилем пользователя.

7. ПО должно поддерживать функцию просмотра расписания во вкладке с профилем пользователя.

8.ПО должно поддерживать функцию просмотра тем рефератов во вкладке с профилем пользователя.

9. ПО должно поддерживать функцию отображения списка группы.

10. ПО должно поддерживать функцию отображения на главном экране последнего просмотренного документа.

11. Каждый документ должен быть подписан:

А) Название;

Б) Загрузочный файл документа;

Г) Автор;

12. ПО должно предусматривать возможность параллельной работы нескольких пользователей с одним и тем же документом.

13. ПО обеспечивает вечное хранение документов.

Словарь предметной области

1. Превью – предварительный просмотр документа;
2. Студия – среда разработки, Android Studio;
3. Прототип – дизайн-макет;
4. Док – документ;

**ПО «NUOS Library»**

***Одностраничное описание системы***

Цель: повышение эффективности работы студентов с обучающей литературой, снижение затрат на обслуживание библиотеки.

Необходимо разработать ПО взаимодействия с обучающей литературой, решающее следующие проблемы:

1. Упрощение взаимодействия студентов с литературой;

2. ПО подсистемы должно удовлетворять нефункциональным требованиям:

A) Весь функционал должен быть протестирован;

Б) Расходы на разработку должны отсутствовать.

3. ПО должно удовлетворять функциональным требованиям:

А) ПО должно поддерживать загрузку интересующих материалов;

Б) ПО должно поддерживать просмотр последних материалов.

4. Необходимо обеспечить возможность "автономной" работы с документами.

5. ПО должно поддерживать многопользовательский режим.

6. ПО должно быть разработано и внедрено за 4 мес.

Выбор методологии проектирования ПО.

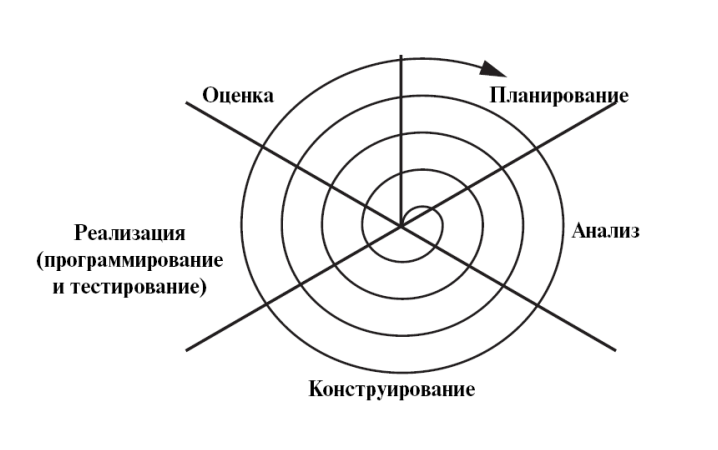
Классическая методология проектирования БД — это мощное направление со своим способами восприятия реальности и существования в ней. В этом течении возникла своя прикладная математика, свое понятие "Предметной Области" (ПрО) и ее моделей. В отношении проектирования БД осознаны и интегрированы в стройные схемы методы выполнения таких проектных этапов:

Классическая модель проектирования:

* Анализ и планирование
* Сбор требований
* Анализ требований
* Планирование проекта
* Проектирование
* Разработка архитектуры
* Разработка моделей данных
* Разработка алгоритмов
* Реализация
* Кодирование
* Отладка
* Тестирование/верификация
* Сопровождение
* Внедрение
* Эксплуатация
* Внесение изменений

Спиральная модель проектирования:

Спирльная модель, предложенная Барри Боэмом в [1986 году](https://ru.wikipedia.org/wiki/1986_%D0%B3%D0%BE%D0%B4), стала существенным прорывом в понимании природы разработки ПО. Она представляет собой [процесс разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), сочетающий в себе как итеративность, так и этапность.



Главная задача — как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований. Основная проблема спирального цикла — определение момента перехода на следующий этап. Для её решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Наша команда выбирала между классической и спиральной моделью. И мы пришли к выбору спиральной, сравнением их. Для нас оказалось крайне важным лёгкость внедрения и тестирования новых функций. Так как в классической модели мы не могли бы вернуться и внедрить дополнительный функционал, хотя для более детального менеджмента времени стоило использовать именно её. Спиральная методология дала возможность качественно отладить все функции. Она вполне подходит для долгосрочных проектов и решений. А так как проект подразумевает долгосрочное сопровождение внедрение нового функционала в последующем сопровождении будет достаточно простым и не будет иметь рисков для программного обеспечения. Исходя из полного состава команды мы можем достаточно быстро проходить полную итерацию проекта. И при необходимости внести коррективы.