# Глоссарий

## I. Категории пользователей

1. **Неавторизованный пользователь** - пользователь, не имеющий доступа к данным внутри системы.
2. **Студент** - авторизованный пользователь, имеющий возможность принимать/отклонять приглашения на тестирование и проходить тестирование, на которое он приглашён.
3. **Преподаватель** - авторизованный пользователь, имеющий возможность создавать тесты и высылать приглашения на тестирование.

## II. Основные термины

1. **Система** - приложение, предназначенное для проведения автоматизированного тестирования студентов, централизованного хранения результатов проведённых тестирований и выдачи этих результатов в визуальной форме пользователям.
2. **Веб-приложение** - совокупность веб-страниц и веб-сценариев, реализующая заданные требования.
3. **Тест** - создаваемая преподавателем совокупность множеств альтернатив тестовых заданий и ограничений на их выполнение.
4. **Тестовое задание** - создаваемый преподавателем текст задания, способ ввода ответа на тестовое задание и ответ на тестовое задание, считающийся правильным.
5. **Множество альтернатив тестовых заданий** - множество тестовых заданий, один элемент которого предлагается выполнить студенту при тестировании.
6. **Ответ на тестовое задание** - некоторая текстовая строка либо упорядоченная совокупность нескольких текстовых строк.
7. **Способ ответа на тестовое задание** - вид элементов управления, позволяющих студенту дать ответ на тестовое задание.
8. **Содержимое теста** - совокупность тестовых заданий из заданного теста, в которой каждое множество альтернатив тестовых заданий представлено одним и только одним элементом.
9. **Чёрный список** - множество пользователей системы категории "преподаватель", которым запрещено отправлять приглашения на тестирование заданному студенту.
10. **Приглашение на тестирование** - гипертекстовое сообщение, приглашающее пользователя категории "студент" к прохождению тестирования и содержащее гиперссылку на веб-страницу, на которой можно пройти тестирование, ФИО преподавателя и учебное заведение, если оно указано.
11. **Данные профиля** - личные данные авторизованного пользователя, хранящиеся в системе. Состав этих данных отражён в пункте 2.1.
12. **Результат тестирования** - упорядоченная совокупность ответов студента на тестовые задания, принадлежащие заданному тесту, с указанием правильности/неправильности каждого ответа.

## III. Функционал

1. **Тестирование** - процесс прохождения теста, включающий в себя решение одного тестового задания из каждого множества альтернатив тестовых заданий.
2. **Решение тестового задания** - процесс сообщения ответа на заданное тестовое задание веб-приложению.
3. **Личный кабинет** - веб-страница или часть веб-приложения, предоставляющая возможность просмотра и редактирования данных профиля авторизованному пользователю.
4. **Экспорт результатов тестирования** (**экспорт**) - формирование электронного документа в формате *xlsx*, содержащего результаты прохождения заданных тестов всеми приглашёнными к их прохождению студентами и выдача гиперссылки на его скачивание преподавателю.
5. **Средство навигации по тестовым заданиям** - набор элементов управления, позволяющих отобразить произвольно выбранное пользователем тестовое задание в рамках проходимого теста.
6. **Элемент управления** - часть графического интерфейса пользователя, позволяющая пользователю осуществлять ввод некоторой информации.

# 1. Требования

## 1.1. Назначение системы

Система тестирования Testmate предназначения для упрощения процесса  
проведения проверочных работ в форме тестирования, а также для  
предоставления единой точки хранения результатов и их анализ.

## 1.2. Цели создания системы

Основными целями создания являются:

1. Помощь преподавателям в организации тестирования студентов
2. Унификация формы тестирования студентов
3. Предоставление возможности создания разнообразных типов тестовых заданий
4. Автоматизация проверки заданий
5. Централизованное хранение результатов выполнения тестов и их предоставление преподавателям и студентам

## 1.3. Задачи системы

Для реализации поставленных целей, система должна решать следующие  
задачи:

1. Размещение веб-приложения, позволяющего создавать тесты и проводить тестирование, по общедоступному адресу в сети Интернет.
2. Создание централизованной базы данных для хранения информации о пользователях, тестах и результатах тестирования.
3. Создание инструментов обработки результатов тестирования

## 1.4. Функциональные требования

1. Личный кабинет
   * В личном кабинете студента отображаются данные его профиля:
     + ФИО
     + Название учебного заведения
     + Курс
     + Факультет
     + Учебная группа
   * В личном кабинете преподавателя отображаются данные его профиля:
     + ФИО
     + Название учебного заведения
   * Личный кабинет преподавателя позволяет ему создать новый тест.
     + Новый созданный тест не содержит тестовых заданий.
     + Новый созданный тест не содержит временного ограничения.
     + Новый созданный тест содержит ограничение по количеству попыток в 1 попытку.
   * В личном кабинете преподавателя отображается список созданных им тестов.
   * Личный кабинет преподавателя позволяет ему приступить к редактированию любого из созданных им тестов.
   * Личный кабинет пользователя предоставляет пользователю возможность редактирования личных данных, которые в нём отображаются.
   * Личный кабинет пользователя предоставляет пользователю возможность изменения его пароля.
2. Тестирование
   * Тестирование (прохождение теста) осуществляется студентом на специальной веб-странице, отображающей содержимое данного теста.
   * Студент может проходить заданный тест ограниченное число раз (ограничение задаётся преподавателем при редактировании теста). Попытка прохождения теста засчитывается при начале прохождения тестирования студентом.
   * Тестирование заключается в последовательном отображении студенту текстов тестовых заданий изсодержимого данного теста и отображении ему средств для ввода ответа на тестовое задание.
   * При каждой попытке прохождения тестирования его содержимое может компоноваться случайным образом.
   * Во время тестирования студент может проходить тестовые задания в произвольном порядке, определяемом самим студентом с помощью формы навигации по тестовым заданиям внутри теста.
   * Выполнение теста может быть ограничено по времени.
     + Временное ограничение задаётся преподавателем при редактировании теста.
     + По истечении временного ограничения студент уведомляется об этом, после чего прохождение тестирования завершается.
   * Независимо от наличия ограничения на время прохождение теста, студент в любой момент тестирования может самостоятельно завершить выполнение теста.
   * После завершения тестирования все тестовые задания, ответ на которые не был дан студентом, считаются неправильно выполненными.
   * Студент дал ответ на тестовое задание, если данное тестовое задание было хотя бы раз выбрано им с помощью средства навигации по тестовым заданиям.
   * Любое изменение средств ввода ответа студентом приводит к запоминанию введённой информации как ответа на тестовое задание.
   * Студент может менять свой ответ на тестовое задание неограниченное число раз во время прохождения тестирования.
3. Способы ответа на тестовое задание
   * Студенту для ввода ответа могут предоставляться элементы управления, позволяющие выполнять ввод данных одним из следующих способов:
     + Выбор одного варианта из заданного множества возможных ответов;
     + Выбор нескольких вариантов из заданного множества ответов;
     + Ввод произвольной текстовой строки;
     + Упорядочивание некоторого множества строк;
     + Установление соответствия между элементами двух множеств.
4. Редактирование теста
   * Редактирование теста осуществляется преподавателем на специальной веб-странице.
   * Веб-страница редактирования теста должна отображать следующую информацию:
     + список принадлежащих данному тесту множеств альтернатив тестовых заданий;
     + список принадлежащих данному тесту тестовых заданий;
     + ограничения по времени и количеству попыток выполнения данного теста.
   * Веб-страница редактирования теста должна позволять выполнять следующие действия:
     + создать новое множество альтернатив тестовых заданий;
     + удалить множество альтернатив тестовых заданий со всем его содержимым;
     + добавить новое задание в множество альтернатив тестовых заданий;
     + редактировать информацию о тексте задания;
     + редактировать способ ответа на тестовое задание;
     + редактировать правильный ответ на тестовое задание;
     + удалить тестовое задание;
     + изменить ограничения на время и количество попыток прохождения данного теста;
     + сохранить все внесённые изменения в системе.
5. Результаты тестирования
   * После завершения тестирования осуществляется определение результата тестирования.
   * Результат тестирования определяется следующим образом: для каждого тестового задания правильность данного студентом ответа определяется по совпадению/несовпадению этого ответа с правильным ответом тестового задания.
   * Система должна хранить и по требованию предоставлять информацию о наилучшей (т.е. с наибольшим количеством правильных ответов) попытке прохождения тестирования студентом.
   * Преподавателю, создавшему тест, доступны результаты тестирования всех студентов, прошедших данный тест.
   * Студенту доступны результаты тестирования для всех пройденных им тестов.
   * Любые результаты тестирования должны быть недоступны преподавателю или студенту, если они не попадают под действие двух предыдущих пунктов.

## 1.5. Нефункциональные требования

1. Система представляет собой веб-приложение.
2. Минимальные требования к браузеру, в котором запускается  
   веб-приложение:
   1. Google Chrome – версии 50.0 или выше
   2. Mozilla Firefox – версии 48.0 или выше
   3. Safari – версии 10.0 или выше
   4. Opera = версии 40.0 или выше
3. Система должна функционировать без перебоев в среднем более, чем 98% от общего периода эксплуатации.
4. Интерфейс системы должен быть наиболее удобен для пользователей, эстетически привлекательным и интуитивно понятным.
5. Ожидаемое время отклика системы зависит от типа и скорости используемого сетевого соединения. При скорости 10 Мбит/с среднее время отклика должно составлять не более 3 секунд.
6. Максимальное время отклика системы - 10 секунд, при превышении данного лимита времени пользователь уведомляется о невозможности подключения к серверу в данный момент.
7. Система предполагает автоматическое обновление веб-приложения раз в N недель.
   1. Пользователь не имеет возможности отказаться от использования обновлённой версии системы.
   2. Обновление веб-приложения осуществляется путём обновления файлов веб-страниц и сценариев, составляющих данное веб-приложение, поэтому каждый раз при обновлении веб-страницы пользователь будет получать её актуальную версию.
8. Архитектура проекта основана на модели трёхзвенного клиент-серверного взаимодействия. Обоснование выбора этой модели и её реализация в рамках системы описана в разделе 6.1.
9. Визуализация UML-диаграмм систем осуществляется с помощью CASE-средства yEd Graph Editor. Обоснование выбора описано в разделе 6.2.
10. Серверный компонент системы реализован на языке Python с использованием фреймворка Django. Клиентский компонент реализован на языке JavaScript.

# 2. Пользователи

## 2.1. Регистрация и авторизация

К регистрации и авторизации допускаются пользователи всех категорий.

Для выполнения регистрации студенту требуется предоставить системе  
следующие данные:

* адрес электронной почты
* ФИО
* Название учебного заведения (опционально)
* Курс (опционально)
* Факультет (опционально)
* Учебная группа
* Пароль, который будет использоваться для авторизации

Для выполнения регистрации преподавателю требуется предоставить системе  
следующие данные:

* Адрес электронной почты
* ФИО
* Название учебного заведения (опционально)
* Пароль, который будет использоваться для авторизации

Для авторизации пользователю необходимо ввести адрес электронной почты и  
пароль, который он задал при регистрации.

## 2.2. Права пользователей

**Неавторизованному пользователю** доступны только регистрация и авторизация.

**Авторизованному студенту** (далее просто **Студенту**) доступны следующие возможности:

1. Деавторизация
2. Потверждение/отклонение приглашения на прохождение тестирования
3. Добавление преподавателя, отправившего приглашение на тестирование, в чёрный список данного студента
4. Прохождение тестирования
5. Просмотр своих результатов тестирования
6. Просмотр и редактирование данных своего профиля

**Авторизованному преподавателю** (далее просто **Преподавателю**) доступны следующие возможности:

1. Деавторизация
2. Создание теста
3. Редактирование теста
4. Удаление теста
5. Отправка приглашений на тестирование
6. Просмотр результатов прохождения студентами одного или нескольких заданных тестирований, созданных данным преподавателем
7. Экспорт результатов тестирования
8. Просмотр и редактирование данных своего профиля

## 2.3. Переход между категориями пользователей

Неавторизованный пользователь может стать авторизованным (**студентом** либо **преподавателем**) при условии успешной авторизации в системе с личными данными аккаунта соответствующего типа.

# 3. Сценарии

## 3.1. Регистрация

**Основной актор**: Неавторизованный пользователь.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Зарегистрироваться в системе.

**Предусловия**: Пользователь не имеет логина и пароля для входа в систему.

**Постусловия**: Пользователь может войти в систему (получает логин и пароль), а данные о нем добавляются в Базу данных (пользователь зарегистрирован).

**Сценарий 1**.

1. Пользователь нажимает кнопку “Зарегистрироваться как студент”.
2. Пользователь заполняет необходимые поля:
   * Адрес электронный почты
   * ФИО
   * Учебная группа
   * Название учебного заведения
   * Пароль, который будет использоваться для авторизации
3. Система осуществляет проверку корректности введенных данных:
   * Адрес электронной почты должен соответствовать стандарту RFC 5322;
   * ФИО не может быть пустой строкой;
   * Учебная группа не может быть пустой строкой;
   * Название учебного заведения не может быть пустой строкой;
   * Пароль не может иметь длину меньше 4 символов.
4. Система записывает введённые пользователем данные в Базу данных, помечая его как неподтверждённого.
5. Система отправляет на указанный адрес электронной почты письмо, содержащее специально сформированную гиперссылку.
6. Пользователь переходит по данной гиперссылке.
7. Система регистрирует переход и помечает данного пользователя как подтверждённого.
8. Система предлагает пользователю авторизоваться под своей учётной записью.

**Сценарий 1a**:

1. Пользователь нажимает кнопку "Зарегистрироваться как преподаватель".
2. Пользователь заполняет необходимые поля:
   * Адрес электронный почты
   * ФИО
   * Название учебного заведения
   * Пароль, который будет использоваться для авторизации
3. Система осуществляет проверку корректности введенных данных:
   * Адрес электронной почты должен соответствовать стандарту RFC 5322;
   * ФИО не может быть пустой строкой;
   * Название учебного заведения не может быть пустой строкой;
   * Пароль не может иметь длину меньше 4 символов.
4. Дальнейшие шаги соответствуют шагам 4.-8. **Сценария 1**.

**Сценарий 1b**:

1. В случае ввода некорректных данных пользователю будет предоставлено описание совершённых им ошибок.

## 3.2. Авторизация

**Основной актор**: Неавторизованный пользователь.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Авторизоваться в системе.

**Предусловия**: Пользователь не авторизован в системе.

**Постусловия**: Пользователь авторизован в системе в качестве студента или преподавателя.

**Сценарий 2**.

1. Пользователь нажимает кнопку "Войти".
2. Пользователь вводит необходимые данные: адрес электронной почты и пароль.
3. Система обращается к Базе данных.
4. База данных осуществляет поиск соотвующей учётной записи пользователя.
5. В случае успеха, Система авторизует пользователя.

**Сценарий 2а**.

1. В случае неуспешного поиска, Система выводит пользователю сообщение о некорректности введённых им данных.

## 3.3. Восстановление пароля

**Основной актор**: Неавторизованный пользователь.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Восстановить пароль учётной записи пользователя.

**Предусловия**: Пользователю известен адрес электронной почты, который он указал при регистрации.

**Постусловия**: Пользователь получает новый пароль для авторизации в системе.

**Сценарий 3**.

1. Пользователь нажимает кнопку "Восстановление пароля".
2. Пользователь вводит адрес электронной почты, который он указал при регистрации.
3. Система обращается к Базе данных.
4. База данных осуществляет поиск соотвующей учётной записи пользователя.
5. В случае успеха, Система генерирует новый пароль и отправляет электронное письмо на указанный адрес электронной почты, содержащее этот пароль.
6. База данных обновляет учётную запись пользователя в соответствии с внесёнными изменениями.
7. Пользователю демонстрируется сообщение о том, что новый пароль был отправлен по указанному им электронному адресу.

**Сценарий 3а**.

1. В случае неуспешного поиска, пользователю демонстрируется сообщение о том, что новый пароль был отправлен по указанному им электронному адресу, однако сообщения по этому электронному адресу не отправляется.

## 3.4. Создание теста

**Основной актор**: Преподаватель.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Создать тест.

**Предусловия**: Преподаватель находится на странице личного кабинета и хочет создать тест.

**Постусловия**: Создаётся новый тест.

**Сценарий 4**.

1. Преподаватель нажимает кнопку "Создать тест".
2. Система отображает диалоговое окно с предложением ввести название теста.
3. Преподаватель вводит желаемое название теста.
4. Система обращается к Базе данных.
5. База данных создаёт запись о новом тесте.
6. Система уведомляет пользователя об успешном завершении операции добавлением созданного теста в список тестов, созданных данным преподавателем.

**Сценарий 4а**.

1. При попытке отправки запроса система обнаруживает отсутствие соединения с сетью.
2. Система демонстрирует пользователю уведомление с описанием ошибки.

## 3.5. Редактирование теста

**Основной актор**: Преподаватель.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Редактировать содержимое теста.

**Предусловия**: Преподаватель находится на странице личного кабинета и хочет изменить содержимое теста.

**Постусловия**: Содержимое теста изменяется.

**Сценарий 5**.

1. Преподаватель нажимает соответствующую нужному тесту кнопку "Редактировать".
2. Система отображает пользователю страницу редактирования теста и отображает ему элементы управления содержимым теста.
3. Пользователь вносит необходимые изменения в содержимое теста с помощью предоставленных ему элементов управления.
4. Пользователь нажимает кнопку "Сохранить изменения".
5. Система проверяет корректность внесённых изменений:
   * Каждое множество альтернатив тестовых заданий должно быть непустым;
   * Каждому тестовому заданию должен быть задан правильный ответ;
   * Текст каждого тестового задания должен быть непустым;
   * Ограничение по времени выполнения теста должно быть неотрицательным числом;
   * Ограничение по количеству попыток выполнения теста должно быть положительным числом.
6. В случае успешной проверки Система обращается к Базе данных.
7. База данных обновляет запись о редактируемом тесте.
8. Система отображает пользователю страницу его личного кабинета и уведомление о том, что изменения успешно сохранены.

**Сценарий 5а**.

1. При проверке корректности внесённых изменений Система обнаружила невыполнение одного или нескольких обозначенных выше пунктов.
2. Система демонстрирует пользователю уведомление, содержащее информацию о всех допущенных им ошибках.

**Сценарий 5b**.

1. Пользователь нажимает кнопку "Отменить изменения".
2. Система запрашивает у пользователя подтверждение этого действия с уведомлением о том, к каким последствиям это может привести.
3. Пользователь подтверждает своё действие.
4. Система отображает пользователю страницу его личного кабинета без модификации записей в Базе данных.

**Сценарий 5с**.

1. Пользователь нажимает кнопку "Отменить изменения".
2. Система запрашивает у пользователя подтверждение этого действия с уведомлением о том, к каким последствиям это может привести.
3. Пользователь отказывается подтвердить своё действие.
4. Система скрывает запрос подтверждения действия, предоставляя пользователю возможность далее вносить изменения в тест.

**Сценарий 5d**.

1. После успешной проверки внесённых изменений на корректность, Система обнаруживает отсутствие соединения с сетью.
2. Система демонстрирует пользователю уведомление с описанием возникшей неполадки.

## 3.6. Удаление теста

**Основной актор**: Преподаватель.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Удалить тест.

**Предусловия**: Преподаватель находится на странице личного кабинета и хочет удалить созданный ранее тест.

**Постусловия**: Тест удаляется из Системы.

**Сценарий 6**.

1. Преподаватель нажимает соответствующую нужному тесту кнопку "Удалить".
2. Система демонстрирует преподавателю уведомление с описанием возможных последствий выполнения данного действия и просит пользователя подтвердить своё намерение.
3. Пользователь подтверждает своё намерение.
4. Система обращается к Базе данных.
5. База данных удаляет соответствующую запись о тесте.
6. Система убирает соответствующую запись о тесте из списка созданных данным преподавателем тестов.

**Сценарий 6а.**

1. Преподаватель отказывается от своего намерения.
2. Система скрывает показанное ранее уведомление.

**Сценарий 6b**.

1. После подтверждения пользователем его намерения, Система обнаруживает отсутствие соединения с сетью.
2. Система демонстрирует пользователю уведомление с описанием возникшей неполадки.

## 3.7. Изменение данных профиля.

**Основной актор**: Авторизованный пользователь.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Изменить данные профиля.

**Предусловия**: Пользователь находится на странице личного кабинета и хочет изменить данные своего профиля.

**Постусловия**: Обновлённые данные профиля сохраняются в Системе.

**Сценарий 7**.

1. Пользователь нажимает кнопку "Редактировать данные", соответствующую одному из следующих полей:
   * Для пользователя категории Студент:
     + ФИО
     + Название учебного заведения
     + Курс
     + Факультет
     + Учебная группа
   * Для пользователя категории Преподаватель:
     + ФИО
     + Название учебного заведения
2. Система демонстрирует пользователю элемент управления, позволяющий изменить выбранное значение.
3. Пользователь вносит изменения в значение выбранного поля и нажимает кнопку "Сохранить изменения".
4. Система проверяет корректность внесённых изменений:
   * ФИО не может быть пустой строкой;
   * Учебная группа не может быть пустой строкой;
   * Название учебного заведения не может быть пустой строкой.
5. При успешной проверке корректности внесённых изменений, Система обращается к Базе данных.
6. База данных обновляет данные соответствующей записи.
7. Система демонстрирует пользователю уведомление об успешном внесении изменений.

**Сценарий 7а**.

1. После внесения изменений в значение выбранного поля пользователь нажимает кнопку "Отменить изменения".
2. Система скрывает элементы управления, позволяющие изменить значение выбранного поля. Внесённые изменения не сохраняются.

**Сценарий 7b**.

1. После проверки изменений на корректность, Система обнаруживает отсутствие соединения с сетью.
2. Система демонстрирует пользователю уведомление с описанием возникшей неполадки.

## 3.8. Изменение пароля

**Основной актор**: Авторизованный пользователь.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Изменить пароль пользователя.

**Предусловия**: Пользователь находится на странице личного кабинета и хочет изменить свой пароль для входа в Систему.

**Постусловия**: Новое значение пароля для входа в Систему заменяет старое.

**Сценарий 8**.

1. Пользователь нажимает кнопку "Изменить пароль".
2. Система запрашивает у пользователя его текущий пароль и значение его нового пароля.
3. Пользователь вводит запрошенные данные и нажимает кнопку "Изменить".
4. Система обращается к Базе данных и проверяет корректность введённых данных:
   * Введённое значение текущего пароля и хранящееся в Базе данных значение пароля пользователя должны совпадать.
   * Длина нового пароля пользователя должна быть не меньше четырёх символов.
5. При успешной проверке, Система запоминает введённое пользователем значение нового пароля, но не заменяет им текущее: пользователь всё ещё должен проходить авторизацию с использованием старого пароля.
6. Система отправляет на электронный адрес пользователя, указанный при регистрации, информирующее о попытке смены пароля электронное сообщение, содержащее специальным образом сформированную гиперссылку.
7. Система демонстрирует пользователю уведомление о том, что для продолжения ему необходимо прочесть указанное электронное сообщение.
8. Пользователь читает указанное электронное сообщение и переходит по специальным образом сформированной гиперссылке в период до 72 часов с момента отправки сообщения.
9. Система определяет переход по гиперссылке и заменяет старое значение пароля новым.
10. Пользователю предлагается авторизоваться с обновлёнными учётными данными.

**Сценарий 8а**.

1. Во время ввода запрашиваемых данных пользователь нажимает кнопку "Отменить".
2. Система скрывает запрос на ввод данных. Внесённые изменения не сохраняются.

**Сценарий 8b**.

1. При неуспешной проверке корректности введённых данных, Система демонстрирует пользователю уведомление с описанием допущенных ошибок.

**Сценарий 8с**.

1. После проверки изменений на корректность, Система обнаруживает отсутствие соединения с сетью.
2. Система демонстрирует пользователю уведомление с описанием возникшей неполадки.

**Сценарий 8d**.

1. Пользователь не переходит по специально сформированной гиперссылке в течение 72 часов.
2. Система удаляет сохранённое значение нового пароля пользователя.
3. Система отправляет по электронному адресу пользователя электронное сообщение, уведомляющее его о недействительности специально сформированной гиперссылки.

**Сценарий 8е**.

1. Пользователь переходит по специально сформированной гиперссылке по истечении 72 часов.
2. Система демонстрирует пользователю веб-страницу с уведомлением, что запрошенной страницы не существует в Системе.

## 3.9. Просмотр результатов тестирования

**Основной актор**: Преподаватель.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Просмотреть результаты тестирования.

**Предусловия**: Преподаватель находится на странице личного кабинета и хочет просмотреть результаты тестирования.

**Постусловия**: Преподавателю демонстрируются результаты тестирования.

**Сценарий 9**.

1. Преподаватель нажимает кнопку "Результаты" соответствующего теста.
2. Система обращается к Базе данных за необходимыми данными.
3. Система демонстрирует преподавателю веб-страницу со всеми результатами прохождения выбранного теста и указанием количества совершённых попыток.

**Сценарий 9а**.

1. Преподаватель нажимает кнопку "Добавить результаты" на странице с результатами некоторого теста.
2. Система обращается к Базе данных для получения списка созданных преподавателем тестов.
3. Система демонстрирует преподавателю список созданных им тестов.
4. Преподаватель выбирает один или несколько тестов из списка и нажимает кнопку "Отобразить".
5. Система обращается к Базе данных за всеми необходимыми результатами тестирования.
6. Система скрывает список созданных преподавателем тестов и демонстрирует преподавателю все результаты прохождения всех выбранных им тестов с указанием количества совершённых попыток.

**Сценарий 9b**.

1. Системе не удаётся получить список созданных преподавателем тестов или результаты прохождения выбранного(-ых) теста(-ов) из-за отсутствия соединения с сетью.
2. Система демонстрирует пользователю уведомление с описанием возникшей неполадки.

**Сценарий 9с**.

1. Преподаватель нажимает кнопку "Экспорт результатов".
2. Система формирует файл в формате *.xlsx* со всеми отображаемыми результатами тестирований.
3. Система предоставляет преподавателю гиперссылку на скачивание файла.
4. По истечении 24 часов система закрывает доступ к файлу и удаляет его.

## 3.10. Отправка приглашений на тестирование

**Основной актор**: Преподаватель.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Отправить приглашение на прохождение тестирования.

**Предусловия**: Преподаватель находится на странице личного кабинета и хочет пригласить студентов на прохождение тестирования.

**Постусловия**: Выбранным студентам отправляются уведомление о приглашении.

**Сценарий 10**.

1. Преподаватель нажимает кнопку "Пригласить" для соответствующего теста.
2. Система запрашивает у преподавателя список адресов электронной почты студентов.
3. Преподаватель вводит все необходимые адреса электронной почты.
4. Система проверяет все электронные адреса на корректность (согласно стандарту RFC 5322).
5. В случае успешной проверки, Система обращается к Базе данных.
6. База данных создаёт записи о новом приглашении на тестировании для каждого электронного адреса из списка.
7. Система отправляет электронные сообщения, содержащие информацию о тесте и гиперссылку на веб-страницу, на которой можно осуществить его прохождение, на каждый адрес электронной почты из списка.
8. Система демонстрирует преподавателю сообщение об успешной отправке приглашений.

**Сценарий 10а**.

1. Система обнаруживает некорректный адрес электронной почты в списке.
2. Преподаватель уведомляется о некорректности введённого адреса электронной почты.
3. Отправка приглашений не производится ни на один из списка указанных адресов.

**Сценарий 10b**.

1. Системе не удаётся обратиться к Базе данных из-за отсутствия соединения с сетью.
2. Система демонстрирует пользователю уведомление с описанием возникшей неполадки.

## 3.11. Принятие приглашения на тестирование

**Основной актор**: Студент.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Принять или отклонить приглашение на прохождение тестирования.

**Предусловия**: Студент находится на странице личного кабинета и хочет принять или отклонить приглашение на тестирование.

**Постусловия**: Студент принимает или отклоняет приглашение на тестирование.

**Сценарий 11**.

1. Студент нажимает кнопку "Принять приглашение" для соответствующего приглашения.
2. Система обращается к Базе данных.
3. База данных удаляет запись о соответствующем приглашении.
4. Система обращается к Базе данных.
5. База данных создаёт запись, означающую осуществление данным студентом 0 попыток прохождения соответствующего теста.
6. Система демонстрирует студенту кнопку "Начать тестирование".

**Сценарий 11а**.

1. Студент нажимает кнопку "Начать тестирование".
2. Система демонстрирует студенту веб-страницу, позволяющую пройти тестирование.

**Сценарий 11b**.

1. Студент нажимает кнопку "Отклонить приглашение".
2. Система обращается к Базе данных.
3. База данных удаляет запись о соответствующем приглашении.
4. Система демонстрирует студенту кнопку "Заблокировать пригласителя".

**Сценарий 11с**.

1. Студент нажимает кнопку "Заблокировать пригласителя".
2. Система обращается к Базе данных.
3. База данных создаёт запись о добавлении преподавателя в чёрный список данного студента.
4. Система демонстрирует студенту уведомление об успешном выполнении действия.

**Сценарий 11d**.

1. Системе не удаётся обратиться к Базе данных из-за отсутствия соединения с сетью.
2. Система демонстрирует студенту уведомление с описанием возникшей неполадки.

## 3.12. Прохождение тестирования

**Основной актор**: Студент.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Пройти тестирование.

**Предусловия**: Студент находится на веб-странице прохождения тестирования.

**Постусловия**: Студент совершает попытку прохождения теста.

**Сценарий 12**.

1. Студент нажимает кнопку "Начать тестирование".
2. Система обращается к Базе данных.
3. База данных находит количество оставшихся попыток на прохождение теста.
4. Система отображает студенту количество оставшихся у него попыток на прохождение теста.
5. Если это количество больше 0, Система обращается к Базе данных.
6. База данных создаёт запись о начале прохождения тестирования студентом с указанием времени начала тестирования.
7. Система отображает студенту текст первого тестового задания из содержимого теста, элементы управления для ввода ответа на тестовое задание, средство навигации по тестовым заданиям и таймер оставшегося времени (в случае наличия ограничения по времени).
8. Студент использует предоставленные ему элементы управления для дачи ответов на тестовые задания.
9. Студент нажимает кнопку "Завершить тестирование".
10. Система демонстрирует студенту уведомление о последствиях его действий и запрашивает подтверждение намерения студента завершить тестирование.
11. Студент подтверждает намерение.
12. Система создаёт результат тестирования путём сверки ответов, данных студентом, с правильными ответами.
13. Система обращается к Базе данных.
14. База данных получает информацию о предыдущем результате тестирования.
15. Система сравнивает количество правильных ответов в предыдущем и текущем результатах тестирования.
16. Система обращается к Базе данных с результатом тестирования, имеющим большее количество правильных ответов.
17. База данных обновляет запись о результате тестирования.
18. Система демонстрирует пользователю веб-страницу с его текущим результатом тестирования.

**Сценарий 12а**.

1. Студент не нажимает кнопку "Завершить тестирование" до истечения ограничения по времени данного теста.
2. Система демонстрирует студенту уведомление о исчерпании лимита времени на прохождение теста.
3. Выполняются пункты 9.-15. **Сценария 12**.

**Сценарий 12b**.

1. Системе не удаётся обратиться к Базе данных из-за отсутствия соединения с сетью.
2. Система демонстрирует студенту уведомление с описанием возникшей неполадки.

**Сценарий 12с**.

1. Если количество оставшихся попыток на прохождение тестирования равно 0, система демонстрирует пользователю уведомление о невозможности прохождения данного теста.

## 3.13. Редактирование чёрного списка

**Основной актор**: Студент.

**Вспомогательный актор**: База данных, Система.

**Контекст использования**: Редактировать чёрный список.

**Предусловия**: Студент находится на веб-странице личного кабинета.

**Постусловия**: Студент внёс необходимые правки в свой чёрный список.

**Сценарий 13**.

1. Студент нажимает кнопку "Редактировать чёрный список".
2. Система обращается к Базе данных.
3. База данных находит все записи о заблокированных студентом преподавателях.
4. Система демонстрирует студенту список заблокированных им преподавателей.
5. Студент нажимает кнопку "Убрать из чёрного списка".
6. Система обращается к Базе данных.
7. База данных удаляет соответствующую запись о заблокированном преподавателе.
8. Система демонстрирует пользователю уведомление о успешном проведении операции удаления.

**Сценарий 13а**.

1. Системе не удаётся обратиться к Базе данных из-за отсутствия соединения с сетью.
2. Система демонстрирует студенту уведомление с описанием возникшей неполадки.

# 4. Модель жизненного цикла

## 4.1 Выбор модели жизненного цикла

В качестве модели жизненного цикла данной системы была выбрана итеративная модель.

Итеративная модель жизненного цикла подразумевает разработку программного обеспечения с линейной последовательностью стадий, но в несколько итераций (версий), т.е. с запланированным улучшением продукта за все время пока Жизненный цикл разработки ПО не подойдет к окончанию.

Разработка программного обеспечения ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах, время жизни каждого из этапов растягивается на весь период разработки.

Жизненный цикл данной модели характерен при разработке сложных и комплексных систем, для которых имеется четкое видение (как со стороны заказчика, так и со стороны разработчика) того, что собой должен представлять конечный результат. Разработка версиями ведется в силу разного рода причин:

* отсутствия у заказчика возможности сразу профинансировать весь дорогостоящий проект;
* отсутствия у разработчика необходимых ресурсов для реализации сложного проекта в сжатые сроки;
* требований поэтапного внедрения и освоения продукта конечными пользователями. Внедрение всей системы сразу может вызвать у её пользователей неприятие и только “затормозить” процесс перехода на новые технологии. Образно говоря, они могут просто “не переварить большой кусок, поэтому его надо измельчить и давать по частям”.

Достоинства модели:

* стабильность требований в течение всего жизненного цикла разработки;
* на каждой стадии формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
* определенность и понятность шагов модели и простота её применения;
* выполняемые в логической последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие ресурсы (денежные. материальные и людские).
* в отличии от каскадной модели существует возможность доработки продукта и возвращение к предыдущим стадиям жизненного цикла

Недостатки модели:

* менеджеры должны постоянно измерять прогресс процесса. в случае быстрой разработки не стоит создавать документы для каждого минимального изменения версии;
* структура системы имеет тенденцию к ухудшению при добавлении новых компонентов — постоянные изменения нарушают структуру системы. Чтобы избежать этого требуется дополнительное время и деньги на рефакторинг. Плохая структура делает программное обеспечение сложным и дорогостоящим для последующих изменений. А прерванный Жизненный цикл ПО приводит еще к большим потерям.

## 4.2. Итерации жизненного цикла

### I итерация

На первой итерации система должна удовлетворять следующим требованиям:

1. Пользователь может зарегистрироваться/авторизоваться в системе
2. Пользователь может деавторизоваться в системе.
3. Пользователю доступен личный кабинет и редактирование личных данных
4. Преподавателю доступно создание теста
5. При редактировании теста преподавателю доступно добавление только одного элемента в множество альтернатив тестовых заданий
6. Преподавателю доступны для создания тестовые задания только со следующими способами ответа:
   * Выбор одного варианта из заданного множества возможных ответов;
   * Выбор нескольких вариантов из заданного множества ответов;
   * Ввод произвольной текстовой строки;
7. Студенту доступно прохождение тестирования.
8. Студенту доступна навигация между заданиями.
9. Во время тестирования студенту доступны к ответу тестовые задания только со следующими способами ответа:
   * Выбор одного варианта из заданного множества возможных ответов;
   * Выбор нескольких вариантов из заданного множества ответов;
   * Ввод произвольной текстовой строки;
10. Преподавателю доступен экспорт данных в формате .csv

### II итерация

На второй итерации система должна удовлетворять всем требованиям **I итерации**, а также следующим требованиям:

1. Преподавателю доступно добавление нескольких элементов во множество альтернатив тестовых заданий.
2. Преподавателю доступны для создания тестовые задания со следующими способами ответа:
   * Упорядочивание некоторого множества строк;
   * Установление соответствия между элементами двух множеств.
3. Студенту доступны для ответа тестовые задания со следующими способами ответа:
   * Упорядочивание некоторого множества строк;
   * Установление соответствия между элементами двух множеств.
4. Преподавателю доступна возможность установить ограничение по времени на прохождение тестирования.

### III итерация

На третьей итерации система должна удовлетворять всем требованиям, описанным в разделе 1.4.

Все дальнейшие итерации будут определены в процессе сопровождения системы.

# 5. Методология разработки программного обеспечения

Для реализации данного проекта была выбрана методология OpenUP.

OpenUP — это итеративно-инкрементальный метод разработки ПО. Позиционируется как легкий и гибкий вариант RUP.

В основу OpenUP положены следующие основные принципы:

* Совместная работа с целью согласования интересов и достижения общего понимания;
* Развитие с целью непрерывного обеспечения обратной связи и совершенствования проекта;
* Концентрация на архитектурных вопросах на ранних стадиях для минимизации рисков и организации разработки;
* Выравнивание конкурентных преимуществ для максимизации потребительской ценности для заинтересованных лиц.

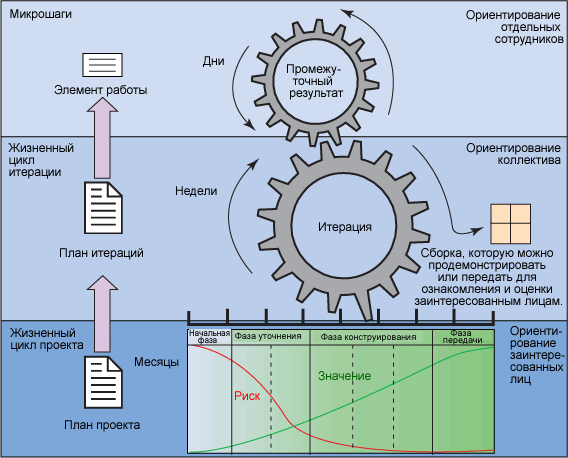


Рис 5.1. Схема методологии OpenUP

OpenUP делит жизненный цикл проекта на четыре фазы: начальная фаза, фазы уточнения, конструирования и передачи. Жизненный цикл проекта обеспечивает предоставление заинтересованным лицам и членам коллектива точек ознакомления и принятия решений на протяжении всего проекта. Это позволяет эффективно контролировать ситуацию и вовремя принимать решения о приемлемости результатов. План проекта определяет жизненный цикл, а конечным результатом является окончательное приложение.

OpenUP делит проект на итерации: планируемые, ограниченные во времени интервалы, длительность которых обычно измеряется неделями. План итерации определяет, что именно должно быть сдано по окончании итерации, а результатом является работоспособная версия. Коллективы разработчиков OpenUP строятся по принципу самоорганизации, решая вопросы выполнения задач итераций и передачи результатов. Для этого они сначала определяют, а затем решают хорошо детализированные задачи из списка элементов работ.

# 6. Архитектура и CASE-средства

## 6.1. Выбор модели архитектуры

Существует несколько видов клиент-серверного взаимодействия, для организации работы данной системы была выбрана трехзвенная архитектура.

Трехзвенная (в некоторых случаях многозвенная) архитектура (N-tier или multi-tier). представляет собой дальнейшее совершенствование технологии "клиент – сервер". Рассмотрев архитектуру " клиент – сервер ", можно заключить, что она является 2-звенной: первое звено – клиентское приложение, второе звено – сервер БД + сама БД. В трехзвенной архитектуре вся бизнес-логика (деловая логика), ранее входившая в клиентские приложения, выделяется в отдельное звено, называемое сервером приложений. При этом клиентским приложениям остается лишь пользовательский интерфейс. В качестве клиентского приложения выступает Web-браузер.

Преимуществом трёхзвенной архитектуры является то, что при изменении бизнес-логики более нет необходимости изменять клиентские приложения и обновлять их у всех пользователей. Кроме того, максимально снижаются требования к аппаратуре пользователей.

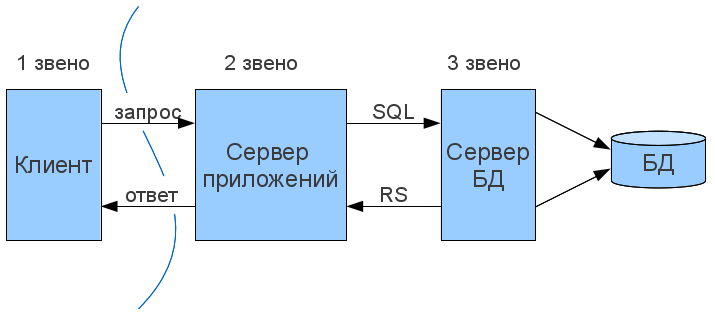


Рис.1 Схема трёхзвенной архитектуры

Взаимодействия между элементами архитектуры построены следующим образом:

* База данных в виде набора файлов находится на жестком диске специально выделенного компьютера (сервера сети).
* СУБД располагается также на сервере сети.
* Существует специально выделенный сервер приложений, на котором располагается программное обеспечение (ПО) делового анализа (бизнес-логика).
* Существует множество клиентских компьютеров, на каждом из которых установлен так называемый "тонкий клиент" – клиентское приложение, реализующее интерфейс пользователя.
* На каждом из клиентских компьютеров пользователи имеют возможность запустить приложение – тонкий клиент. Используя предоставляемый приложением пользовательский интерфейс, он инициирует обращение к ПО делового анализа, расположенному на сервере приложений.
* Сервер приложений анализирует требования пользователя и формирует запросы к БД. Для общения используется специальный язык запросов SQL, т.е. по сети от сервера приложений к серверу БД передается лишь текст запроса.СУБД инкапсулирует внутри себя все сведения о физической структуре БД, расположенной на сервере.
* СУБД инициирует обращения к данным, находящимся на сервере, в результате которых результат выполнения запроса копируется на сервер приложений.
* Сервер приложений возвращает результат в клиентское приложение (пользователю).
* Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов.

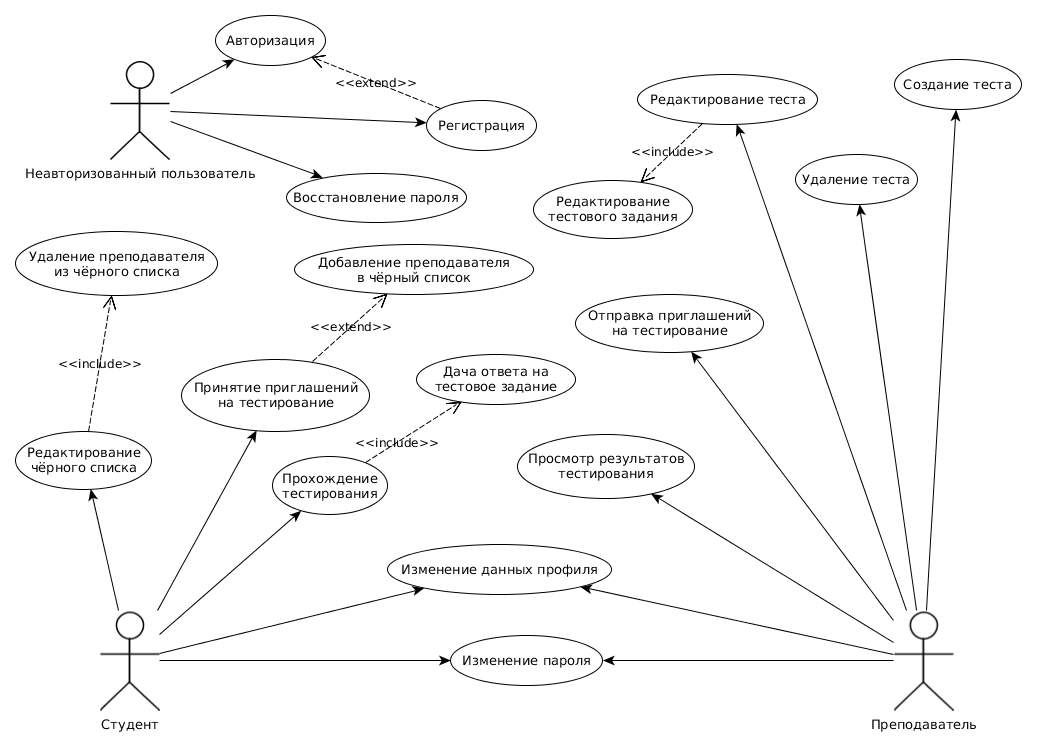
## 6.2. Выбор CASE-средств

В качестве CASE-средства моделирования при проектировании данной системы было решено использовать yEd Graph Editor. Данное программное обеспечение является бесплатным, но так как оно является проприетарным распространяется в виде бинарных дистрибутивов для различных платформ, которые можно скачать с официального сайта продукта (<https://www.yworks.com/products/yed>).

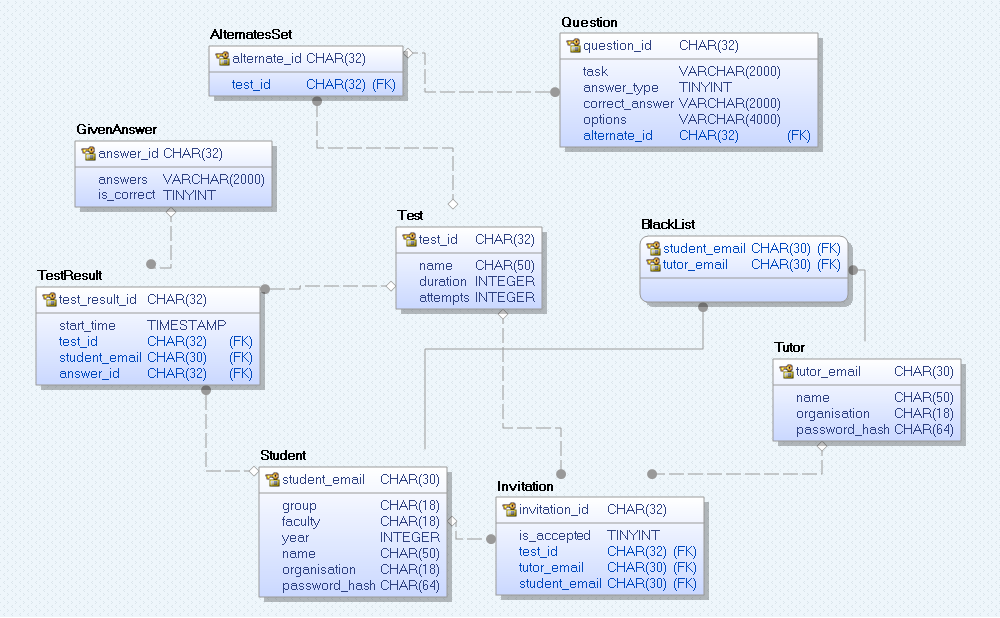
В качестве ключевых особенностей данного продукта можно выделить:

* Интуитивный, легко настраиваемый интерфейс для построения диаграм
* Данное ПО поставляется с обширной палитрой готовых к использованию элементов, в том числе:
* Наличие дистрибутивов под все основные платформы (Windows, Linux, MacOS)
* Автоматическая реорганизация диаграмм
* Простой импорт данных из файлов различных форматов:
  + XML
  + XLS
  + GML
  + GEDCOM
* Экспорт диаграмм в множество различных форматов:
  + Adobe® Portable Document Format (PDF)
  + Adobe® Flash® (SWF)
  + SVG, который поддерживает встроенные ссылки и описательные всплывающие подсказки
  + JPG, GIF, BMP, PNG. Большие диаграммы могут быть экспортированы в виде нескольких изображений.
  + HTML image maps с поддержкой вложенных URL-адресов и подсказок.

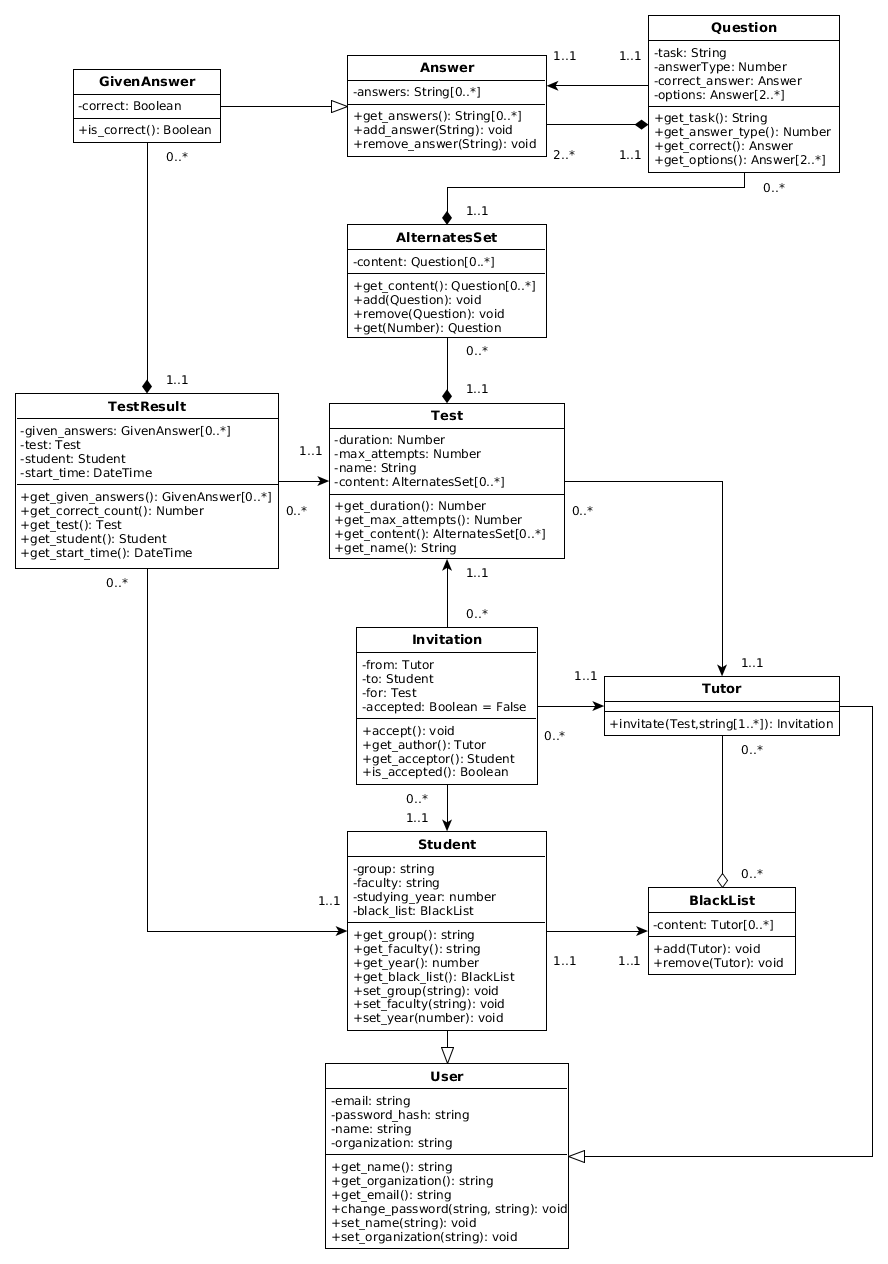
# 7. USE-CASE диаграмма



# 8. Физическая модель базы данных



# 9. UML-диаграмма классов



# 10. Тестирование программного продукта

## 10.1. Выбор модели тестирования

При выборе методики тестирования основной задачей являлось максимально быстрое и наименее затратное получение работоспособного прототипа системы. Так как в качестве модели жизненного цикла была выбрана итерационная модель и методология OpenUP, было принято решение использовать регрессионное тестирование, придерживаясь стратегий черного и белого ящиков. Также будут использоваться системные и стресс- тесты.

Регрессионные тесты позволят проверить изменения, сделанные в приложении или окружающей среде, для подтверждения того факта, что существующая ранее функциональность работает, как и прежде. При этом несомненным преимуществом данного метода является быстрое выявление ошибок на каждом этапе разработки. Данное тестирование позволит эффективно тестировать общую производительность системы и совместную работу всех ее компонентов, а также работу нововведенных модулей на каждой итерации разработки, в том числе и финальной.

Для регрессионного тестирования будут использоваться тест кейсы, написанные на ранних стадиях разработки и тестирования. Это даст гарантию того, что изменения в новой версии приложения не повредили уже существующую функциональность. В качестве инструмента тестирования будет использован SimpleTest, который поддерживает фиктивные объекты и может быть использован для автоматизации регрессионного тестирования. Также для автоматизации системного тестирования будет использовано ПО Selenium.

## 10.2. Тест-план

### 10.2.1. Цель

Целью составления данного тест плана является описание процесса тестирования системы тестирования Testmate. Документ позволяет получить представление о мероприятиях по тестированию приложения.

### 10.2.2. Стратегия процесса тестирования

**I этап: регрессионное тестирование**

При прохождении данного этапа тестирования планируется выявить ошибки полученные в связи с добавленными изменениями на каждой итерации, основанные на взаимодействии функционалов путем реализации нетривиальных сценариев.

**II этап: системное тестирование**

Этот этап основан на сценариях (корректных и некорректных) работы пользователя с системой, описывающих полный цикл работы с системой.

**III этап: стресс-тестирование, нагрузочное тестирование**

На данном этапе определяются уязвимости и возможность работы приложения под нагрузкой.

### 10.2.4. Типы тестирования

10.2.4.1. Регрессионное тестирование

Цель: выявить функциональные ошибки в основанных на взаимодействии модулей и в отдельных модулях путем реализации нетривиальных сценариев.  
Описание процесса тестирования:

При каждом изменении проводятся следующие проверки:

* Регрессия багов (Bug regression) - попытка доказать, что исправленная ошибка на самом деле не исправлена;
* Регрессия старых багов (Old bugs regression) - попытка доказать, что недавнее изменение кода или данных сломало исправление старых ошибок, т.е. старые баги стали снова воспроизводиться;
* Регрессия побочного эффекта (Side effect regression) - попытка доказать, что недавнее изменение кода или данных сломало другие части разрабатываемого приложения.

Таким образом, на каждом этапе проводится проверка как нововведенных изменений, так и старых модулей, а также всего приложения целиком.

10.2.4.2. Системное тестирование

Цель: убедиться, что пользователь может использовать основной функционал приложения в процессе реального использования. Проверить степень удобства приложения для пользователя на основании общих критериев и субъективного восприятия. Проверить корректную работу и дизайн приложения в различных системах (браузерах и ОС) и при различных разрешениях монитора, в том числе и в мобильных версиях.

Описание процесса тестирования:

Запуск приложения и воспроизведение сценариев пользователя в нем в различных средах и браузерах.

10.2.4.3. Стресс-тестирование

Цель: выявить уязвимости в работе приложения путем составления модели нагрузки  
Данный вид тестирования и тест-кейсы к нему оговариваются и описываются отдельно вместе с разработчиками приложения.

## 10.3. Итоги

Итогом тестирования будет являться отчёт, отражающий все выявленные проблемы, а также включающих в себя набор пользовательских пожеланий.

# 11. План сопровождения продукта

## 11.1. Концепция сопровождения

### Типы сопровождения

Предполагается использовать сопровождения следующих типов:

* адаптивное (модификация ПО в условиях изменяющейся среды);
* профилактическое (обнаружение и исправление скрытых ошибок в ПО);
* корректирующее (реактивное изменение ПО для корректировки обнаруженых проблем);

### Организация техподдержки

Пользователи системы имеют возможность вести e-mail переписку со службой технической поддержки системы. E-mail техподдержки указывается на каждой веб-странице приложения. Специалисты рассматривают вопросы, связанные с:

* Общими сведениями по эксплуатации системы;
* Обнаружением и устранением неполадок системы;
* Предложениями по улучшению и развитию системы.

## 11.2. Практическое применение процесса сопровождения

Процесс сопровождения состоит из следующих задач:

* устранение неполадок;
* улучшение дизайна;
* расширение функциональности;
* вывод программного обеспечения из эксплуатации.

## 11.3. Причины необходимости сопровождения

1. Необходимость в обновлении применяемых технологий, связанная с вероятным моральным устареванием актуальных на данный момент технологий.
2. Возможное наличие скрытых ошибок в системе.
3. Возможное снижение качества оказания облачных услуг.
4. Изменение других, неучтённых внешних факторов.

## 11.5. Исполнители данной работы:

* Команда технических писателей - модифицирует требования и документацию.
* Команда проектировщиков - модификацирует архитектуру системы.
* Команда разработчиков - занимается рефакторингом и модификацией кода.
* Команда тестировщиков - тестирует новые версии программного продукта.
* Менеджеры по организации сопровождения - занимаются общими вопросами по организации сопровождения.

В случае возникновение неполадок прикладного (программного) характера ответственность накладывается на разработчиков и тестировщиков данного программного продукта (системы).

## 11.6. Форма выполнения работ

1. Выход свежей версии (Модификация). Разрабатывается вне зависимости от работы уже сопровождаемой системы. В зависимости от степени «обновления» внедряется в уже существующую систему или заменяет существующую.
2. Устранение неполадок. Выполняется в непосредственной связи с системой.
3. Вывод из эксплуатации.

## 11.7. Сроки сопровождения

Сопровождение будет гарантированно осуществляться весь период до вывода системы из эксплуатации.