ФИО: Курнаев Данила Владимирович
Группа: М7О-606С-19
Дата сдачи: 09.12.2024

Тема: «Обеспечение безопасных условий труда при разработки системы эргономической оценки кабины самолета на основе теста психомоторной бдительности, методики PVT и NASA-TLX.».

Оглавление

Обозначения

Термины и определения

NASA-TLX (NASA Task Load Index) - это широко используемый субъективный многомерный инструмент оценки рабочей нагрузки, разработанный в 1980-х годах в Исследовательском центре Эймса НАСА.

PVT - тестирование (Psychomotor Vigilance Task) — это метод оценки психомоторной бдительности и скорости реакции человека на визуальные стимулы.

Летный экипаж – персонал, управляющий летательным аппаратом в полете.

Стека технологий – это совокупность программных инструментов, библиотек, языков программирования и фреймворков, используемых для разработки программного обеспечения.

ПО (Программное обеспечение) – это набор программ и сопутствующей документации, предназначенный для выполнения определённых задач на компьютерах и других вычислительных устройствах.

ORM (Object-Relational Mapping) — метод программирования, позволяющий преобразовывать данные между несовместимыми типами систем с использованием объектно-ориентированного программирования. ORM упрощает взаимодействие с базами данных, превращая запросы в объекты языка программирования.

MVP (Minimum Viable Product) — минимально жизнеспособный продукт, представляющий собой первую рабочую версию приложения, содержащую только базовую функциональность. Основная цель MVP — быстрое тестирование идеи с минимальными затратами.

SQLite — лёгкая реляционная база данных, которая хранит все данные в одном файле. Её особенности включают простоту интеграции, автономность и кроссплатформенность.

SQLAlchemy — популярная библиотека для Python, предоставляющая инструменты для работы с базами данных через ORM и прямые SQL-запросы.

PyInstaller — инструмент для создания исполняемых файлов из Python-программ. Он поддерживает кроссплатформенность и упрощает распространение приложений.

Pydantic — библиотека Python для валидации данных и работы с типами. Использует аннотации типов Python для проверки и сериализации данных.

Pattern Matching — новая возможность Python 3.10, позволяющая использовать синтаксис, аналогичный конструкции switch/case, для обработки сложных данных и упрощения логики программирования.

Введение

Современная авиация предъявляет высокие требования к эргономике и эффективности человеко-машинного взаимодействия. Кабина пилота, как ключевой элемент авиационной техники, должна обеспечивать оптимальные условия для выполнения задач в различных режимах эксплуатации. От эргономических характеристик кабины зависят не только комфорт и здоровье пилотов, но и безопасность полетов в целом.

Эргономическая оценка кабины самолета требует комплексного подхода, включающего анализ когнитивной нагрузки, психофизиологических особенностей пилота, а также использование передовых технологий тестирования.

Методы тестирования психомоторной бдительности (PVT) и методика NASA-TLX доказали свою эффективность в оценке функционального состояния операторов. PVT позволяет объективно измерять когнитивные способности и реакцию оператора в реальном времени, а NASA-TLX — выявлять субъективное восприятие нагрузки по различным параметрам, таким как психическое напряжение, усилия и временные ограничения. Интеграция этих методов в единую систему оценки кабины самолета открывает новые возможности для повышения точности анализа и улучшения эргономических характеристик авиационной техники.

Целью данной работы является разработка системы эргономической оценки кабины самолета, объединяющей тесты PVT и методику NASA-TLX. Для достижения этой цели было создано программное обеспечение, обеспечивающее автоматизацию тестирования, сбор и обработку данных, а также визуализацию результатов. В процессе работы проведен обзор существующих методов эргономической оценки человеко-машинных интерфейсов, разработана методика анализа состояния оператора и представлено графическое описание созданного приложения.

1. История и эволюция оценки эргономики в авиации

1.1. Основные этапы развития эргономики в авиации.

Эргономика в авиации начала развиваться с первых лет авиационной индустрии, когда стало понятно, что комфорт и удобство пилота влияют на безопасность полетов. В 1930-х годах начались первые исследования по проектированию кабины пилота с учетом физиологических и когнитивных особенностей человека. Основной акцент делался на улучшение видимости приборов и эргономичное расположение органов управления.

После Второй мировой войны эргономика получила новый импульс благодаря развитию реактивной авиации. Быстрые самолеты требовали большей концентрации и реакции пилотов, что привело к созданию более интуитивных приборных панелей и удобных кресел. В 1960-х годах стали активно применяться принципы человеко-машинного взаимодействия, включая первые методы оценки когнитивной нагрузки.

В 1980–1990-х годах началось активное использование компьютерного моделирования для проектирования кабины пилота. Это позволило тестировать эргономические решения без необходимости строить физические макеты. С этого момента эргономика стала неотъемлемой частью проектирования всех современных самолетов.

1.2. Уменьшение численности экипажей: от многоместных экипажей до двух пилотов.

Первоначально управление самолетами требовало большой команды, включающей пилота, штурмана, радиста и инженера. Каждая из этих ролей была необходима для обеспечения надежности полета. Однако с развитием технологий численность экипажа постепенно уменьшалась.

В 1970-х годах внедрение автоматических систем навигации и управления двигателями позволило сократить экипаж до трех человек, исключив роль штурмана. К 1990-м годам с появлением цифровых систем управления и усовершенствованных автопилотов экипаж уменьшился до двух пилотов. Эти изменения повысили требования к когнитивным и

физическим возможностям пилотов, что сделало эргономику еще более важной частью проектирования кабины.

1.3. Особенности современной эргономики в кабине пилота.

Современные кабины самолетов проектируются с учетом следующих факторов:

- Интуитивность управления: Все органы управления расположены так, чтобы минимизировать время на их использование.
- Интеграция систем: Информация с приборов объединяется на мультифункциональных дисплеях, что облегчает восприятие данных.
- Когнитивная поддержка: Используются системы предупреждений и подсказок, снижающие нагрузку на пилотов в стрессовых ситуациях.
- Эргономичные кресла: Удобные сиденья с поддержкой поясницы и возможностью регулировки положения минимизируют физическую усталость во время длительных рейсов.

Эти элементы направлены на обеспечение безопасности и эффективности работы пилотов, особенно в условиях длительных перелетов и высоких нагрузок.

2. Современная нагрузка на пилотов

Добавлено примечание ([EDG1]): Что это???

2.1. Факторы когнитивной и физической нагрузки на пилотов.

Работа пилотов сопряжена с высокой когнитивной и физической нагрузкой, особенно во время взлета, посадки и работы в сложных погодных условиях. Основные факторы включают:

- Когнитивная нагрузка: Обработка большого объема информации, необходимость быстрого принятия решений в условиях ограниченного времени.
- Физическая нагрузка: Усталость от длительного нахождения в одном положении, особенно на длительных рейсах.
- Психоэмоциональный стресс: Ответственность за жизнь пассажиров и экипажа.

Исследования показывают, что высокая когнитивная нагрузка может приводить к ошибкам, связанным с человеческим фактором, что делает использование эргономичных решений жизненно необходимым.

2.2. Анализ инцидентов, связанных с человеческим фактором.

Согласно отчету Международной организации гражданской авиации (ICAO), до 70% авиационных происшествий вызваны ошибками пилотов. [1].

Добавлено примечание ([EDG2]): Источник?

Наиболее распространенные причины включают:

- Усталость, связанная с длительным временем полета.
- Неправильное восприятие информации с приборов.
- Ошибки, вызванные перегрузкой информации.

Примером является инцидент 2009 года с рейсом Air France 447, где ошибки пилотов в условиях сложной погодной ситуации привели к катастрофе. Этот случай подчеркивает важность разработки инструментов для оценки и снижения когнитивной нагрузки.

2.3. Методы измерения нагрузки: применение PVT и NASA-TLX.

PVT (Psychomotor Vigilance Test) и NASA-TLX (Task Load Index) являются одними из ключевых инструментов для оценки нагрузки на

пилотов, но не единственными в этой области. Хотя эти методы играют важную роль в исследовании когнитивной и физической нагрузки, они чаще используются как часть более обширных систем анализа. Кроме того, их совокупное применение для оценки эргономики в авиации пока не является стандартной практикой и не всегда охватывает все аспекты взаимодействия человека и системы.

PVT: используется для измерения уровня бодрствования и времени реакции. Этот тест помогает оценить усталость пилотов и их способность к выполнению задач в сложных условиях.

NASA-TLX: позволяет определить субъективную рабочую нагрузку, включая умственную, физическую и временную нагрузки. Это помогает адаптировать рабочие условия и интерфейсы кабины к возможностям пилотов.

Эти методы применяются как в авиации, так и в других отраслях для анализа и оптимизации рабочих процессов. Их использование способствует снижению числа ошибок и повышению уровня безопасности полетов.

3. Теоретические основы эргономической оценки

3.1. Основы эргономики в авиации

Эргономика — это научная дисциплина, изучающая взаимодействие человека с элементами системы с целью оптимизации производительности и обеспечения комфорта, безопасности и эффективности труда. В авиации эргономика направлена на адаптацию рабочих мест, оборудования и процедур к возможностям и ограничениям человека, учитывая физические, когнитивные и организационные аспекты деятельности пилотов и экипажа.

Кабина пилота является сложной человеко-машинной системой, где эргономические аспекты напрямую влияют на безопасность и эффективность полетов. Неправильное расположение приборов, неудобные органы управления или избыточная когнитивная нагрузка могут привести к ошибкам пилотирования. Исследования показывают, что эргономические недостатки кабин самолетов являются факторами риска для безопасности полетов, подчеркивая необходимость тщательной эргономической проработки кабин разрабатываемых самолетов.

3.2. Методы оценки эргономических характеристик

Существует множество методов оценки эргономических характеристик в авиации, включая аналитические, экспериментальные и экспертные подходы. Аналитические методы основываются на математическом моделировании и прогнозировании взаимодействия человека с системой. Экспериментальные методы включают лабораторные и полевые исследования с участием операторов. Экспертные методы предполагают использование опросников, которые участники заполняют самостоятельно, фиксируя свои ощущения и данные о нагрузке. Это позволяет собрать первичную информацию, отражающую субъективное восприятие параметров, таких как усталость, стресс и когнитивная нагрузка. Затем эти данные передаются экспертам, которые проводят их детальный анализ и оценку. Такой подход сочетает в себе преимущества самофиксации, обеспечивающей непосредственный доступ к индивидуальному восприятию, и экспертной интерпретации,

позволяющей учитывать объективные аспекты и делать выводы о состоянии оператора и эргономических характеристиках системы. Комбинация этих подходов позволяет получить всестороннюю оценку эргономики авиационных систем.

Человеко-машинное взаимодействие в авиации характеризуется высокой сложностью и динамичностью. Пилоты должны быстро и точно обрабатывать большие объемы информации, принимая решения в условиях временного давления и стресса. Эргономические исследования человеческого фактора в современных технических системах подчеркивают необходимость согласования конструктивных особенностей машин с характеристиками деятельности человека-оператора для повышения надежности и безопасности эксплуатации.

3.3. Теоретические основы тестов PVT и NASA-TLX

3.1.1. Тест психомоторной бдительности (PVT)

Тест психомоторной бдительности (PVT) представляет собой объективный инструмент для измерения уровня бодрствования и когнитивной функции оператора. Этот тест разработан для оценки скорости реакции на визуальные стимулы, что делает его идеальным для определения снижения бдительности, вызванного усталостью, стрессом или другими факторами.

3.3.1.1. Принципы работы PVT

PVT основывается на простой задаче: пользователь должен как можно быстрее нажимать кнопку в ответ на появление визуального сигнала на экране. Записываются время реакции и количество пропущенных стимулов, что позволяет определить уровень усталости или снижение когнитивной функции. Основные показатели включают:

- Среднее время реакции.
- Частота ошибок (пропущенные сигналы).
- Вариабельность времени реакции.

3.3.1.2. Алгоритм проведения теста PVT

Подготовка: Устройство для тестирования (компьютер, планшет или специализированное оборудование) должно быть настроено на генерацию визуальных стимулов с нерегулярными интервалами.

Тестирование: Пользователь проходит серию испытаний, реагируя на визуальные стимулы. Время реакции фиксируется автоматически.

Анализ: Полученные данные обрабатываются для вычисления средних значений и выявления отклонений.

Интерпретация: Результаты анализируются с учетом индивидуальных характеристик пользователя и условий тестирования.

3.3.1.3. Применение PVT

PVT широко используется в авиации для оценки готовности пилотов к выполнению сложных задач, особенно в условиях длительных рейсов. Например, тестирование пилотов перед сменой позволяет определить уровень их усталости и принять соответствующие меры для повышения безопасности полетов. Также PVT активно применяется в медицине, транспорте и военной сфере.

3.1.2. NASA Task Load Index (NASA-TLX)

NASA-TLX (NASA Task Load Index) — это широко используемый субъективный многомерный инструмент оценки рабочей нагрузки, разработанный в 1980-х годах в Исследовательском центре Эймса НАСА. Он предназначен для оценки воспринимаемой оператором рабочей нагрузки при выполнении различных задач и используется в таких областях, как авиация, здравоохранение и других сложных социотехнических системах.

3.3.2.1. Структура NASA-TLX

NASA-TLX включает шесть шкал, каждая из которых оценивается пользователем по семибалльной шкале:

Ментальная нагрузка: степень умственной и перцептивной активности, необходимой для выполнения задачи (например, мышление, принятие решений, запоминание).

Физическая нагрузка: объем физической активности, требуемой для выполнения задачи (например, нажатие кнопок, управление устройствами).

Временная нагрузка: ощущаемое давление времени, связанное с выполнением задачи.

Усилие: количество усилий, затраченных для достижения уровня производительности в задаче.

Производительность: субъективная оценка успешности выполнения задачи и удовлетворенности результатом.

Уровень фрустрации: степень раздражения, стресса и неудовлетворенности, испытываемых во время выполнения задачи.

3.3.2.2. Алгоритм проведения NASA-TLX

Подготовка: Участнику объясняются шкалы и процедура оценки.

Оценка: После выполнения задачи участник оценивает каждый из шести параметров.

Взвешивание: Участник распределяет вес значимости между параметрами, что позволяет учитывать индивидуальные особенности задачи.

Анализ: Рассчитывается общий индекс нагрузки путем усреднения взвешенных значений.

3.3.2.3. Применение NASA-TLX

PVT широко используется в авиации для оценки готовности пилотов к выполнению сложных задач, особенно в условиях длительных рейсов. Например, тестирование пилотов перед сменой позволяет определить уровень их усталости и принять соответствующие меры для повышения безопасности полетов. Также PVT активно применяется в медицине, транспорте и военной сфере.

3.1.3. Сравнение и сочетание методов PVT и NASA-TLX

Оба метода, PVT и NASA-TLX, часто используются в сочетании для получения полного анализа состояния оператора:

- **PVT**: Обеспечивает объективную оценку когнитивной функции, измеряя скорость реакции.
- **NASA-TLX**: Позволяет понять субъективное восприятие нагрузки пользователем.

Их совместное использование особенно полезно в авиации, где необходимо учитывать как объективные показатели, так и субъективные ощущения пилотов. Например, пилот может демонстрировать нормальное время реакции по PVT, но сообщать о высоком уровне фрустрации и умственной нагрузки по NASA-TLX, что указывает на необходимость адаптации условий работы или интерфейса.

4. Оценка эргономики в различных отраслях

Эргономика давно вышла за рамки традиционных производственных процессов и сегодня охватывает множество отраслей, где человек взаимодействует с техникой, информацией или физической средой. Ее принципы применяются для повышения безопасности, производительности и удовлетворенности пользователей, адаптируя системы и процессы к человеческим возможностям и ограничениям.

В каждой отрасли эргономика сталкивается с уникальными задачами. Несмотря на различия, объединяющим элементом является стремление обеспечить гармоничное взаимодействие между человеком и системой, минимизируя риски и оптимизируя результаты.

В этом разделе мы подробно рассмотрим, как принципы эргономики реализуются в нескольких ключевых сферах, таких как медицина, IT, транспорт и военная индустрия. Для каждой из отраслей будет представлено влияние эргономики, решаемые задачи и достигнутые результаты.

4.1. Эргономика в медицинской сфере

Эргономика в медицине играет ключевую роль в обеспечении безопасности труда медперсонала и улучшении качества лечения пациентов. В последние годы этот аспект стал особенно актуальным, поскольку медицинская сфера сталкивается с новыми вызовами, такими как повышенная нагрузка на врачей и медсестер. Интересно, что многие решения в этой области разрабатываются на основе данных, полученных из реальных медицинских учреждений и научных исследований.

4.1.1. Ключевые задачи медицинской эргономики

Медицинская эргономика направлена на создание условий труда, которые минимизируют физическую и психическую нагрузку на медицинский персонал. В условиях высокой рабочей нагрузки современные медицинские учреждения активно применяют научные подходы, включая использование специализированного программного обеспечения для анализа эргономических параметров. Эти методы позволяют минимизировать

физические и когнитивные нагрузки, а также предотвратить профессиональные заболевания.

Особое внимание уделяется профилактике профессиональных заболеваний, таких как остеохондроз позвоночника, который часто встречается у медсестёр из-за физических перегрузок при перемещении пациентов и оборудования.

4.1.2. Задачи медицинской эргономики

Основные задачи медицинской эргономики включают:

- **Оптимизацию операционных пространств:** Применение эргономичных операционных столов с автоматической регулировкой высоты и угла наклона способствует снижению утомляемости хирургов.
- **Разработку удобных инструментов:** Антропометрическое проектирование хирургического оборудования снижает физическую нагрузку и повышает точность манипуляций.
- **Реорганизацию рабочих мест медсестер:** Удобное размещение оборудования и материалов уменьшает риск травм и ошибок.

4.1.3. Оценка когнитивной нагрузки

Методика NASA-TLX применяется для оценки когнитивной нагрузки медицинского персонала, особенно в ситуациях, требующих высокой концентрации внимания. Исследования показали, что внедрение эргономичных решений позволяет снизить уровень стресса на 20%, что ведет к повышению скорости принятия решений на 15% в условиях высокой интенсивности работы.

4.1.4. Примеры внедрения

Примером успешного применения является проектирование операционных в клиниках Германии. Использование эргономичных столов и оптимизация расположения оборудования позволили сократить длительность операций на 10–15%. Такие решения снижают вероятность

профессионального выгорания у хирургов и повышают общее качество медицинской помощи.

4.2. IT и офисная среда

Эргономика в IT и офисной среде направлена на улучшение физического и когнитивного комфорта сотрудников. Современные решения включают как физические улучшения рабочего пространства, так и использование цифровых инструментов для оценки и оптимизации эргономических характеристик.

4.2.1. Физическое пространство

Регулируемые по высоте столы, эргономичные кресла и специализированное освещение стали стандартом в крупных IT-компаниях. Такие улучшения снижают риск профессиональных заболеваний, связанных с длительным пребыванием в одной позе. Например, сотрудники, использующие регулируемые столы, демонстрируют увеличение производительности труда на 12%.

4.2.2. Программное обеспечение

Добавлено примечание ([EDG3]): В IT есть интересные решения с айтреккером

Эргономика программного обеспечения фокусируется на разработке удобных интерфейсов, минимизирующих когнитивную нагрузку. Системы анализа, такие как NASA-TLX, позволяют разработчикам оптимизировать цифровые продукты, обеспечивая удобство пользователей. Результаты исследований показывают, что интуитивные интерфейсы сокращают время выполнения задач на 30%.

Примером использования таких методов является применение программного обеспечения для отслеживания движений глаз, такого как iTracker. Этот инструмент позволяет анализировать поведение пользователя, выявлять сложные участки интерфейса и оптимизировать их для улучшения пользовательского опыта. Анализ с использованием iTracker может дополнить данные, полученные от опросников, таких как NASA-TLX, предоставляя разработчикам объективную информацию о нагрузке на пользователей при взаимодействии с системой.

4.2.3. Примеры внедрения

Компании Google и Microsoft внедряют эргономические рабочие пространства и инструменты для анализа когнитивной нагрузки, что позволяет сократить количество ошибок в работе сотрудников и снизить уровень профессионального выгорания. Эти меры привели к сокращению числа больничных на 20%.

Добавлено примечание ([EDG4]): Ссылка на источник?

4.3. Автомобильная промышленность

Эргономика в автомобильной промышленности направлена на повышение комфорта и безопасности как для водителей, так и для пассажиров. Особое внимание уделяется проектированию водительского места, а также системам оценки и управления когнитивной нагрузкой водителей.

4.3.1. Водительское место

Внедрение эргономичных кресел с поддержкой поясницы и автоматической регулировкой положения позволяет снизить утомляемость водителей на длительных маршрутах. Современные интерфейсы управления помогают водителю быстрее адаптироваться к транспортному средству, что повышает уровень безопасности.

4.3.2. Оценка внимания и усталости

Системы предупреждения усталости водителя разрабатываются на основе тестов PVT. Эти системы мониторят внимание и состояние водителя в режиме реального времени, отправляя сигналы о необходимости отдыха. Применение таких решений, значительно, снижает риск ДТП.

Добавлено примечание ([EDG5]): Это как? Очень сомнительно. На PVT сложно создать систему мониторинга, потому что это тест, а не параметр мониторинга. Там ставят камеры Возможно, браслеты.

4.3.3. Примеры внедрения

Автомобили Tesla и Volvo оснащены адаптивными креслами и интеллектуальными системами мониторинга усталости водителей. Это способствует снижению аварийности и увеличению общей удовлетворенности клиентов. Эргономические улучшения также положительно влияют на продажи и имидж производителей.

4.4. Военная сфера

Эргономика в военной сфере охватывает проектирование рабочих мест операторов, кабин пилотов и систем управления, обеспечивая эффективность действий в экстремальных условиях. Разработка военной техники включает как физические, так и когнитивные аспекты взаимодействия человека с оборудованием.

4.4.1. Кабины летчиков

Современные кабины истребителей разрабатываются с учетом эргономических требований. Интуитивно понятное расположение органов управления, минимизация вибрации и улучшенная обзорность повышают точность и скорость действий пилотов в сложных условиях.

4.4.2. Когнитивная нагрузка

Применение NASA-TLX позволяет разрабатывать интерфейсы и оборудование, которые минимизируют стресс и когнитивные перегрузки операторов. Это особенно важно для обеспечения безопасности при выполнении миссий в условиях высокой напряженности.

4.4.3. Примеры внедрения

Примером является внедрение модернизированных кабин для военных истребителей, что позволило сократить время реакции пилотов на 15%. Такие улучшения способствуют успешному выполнению миссий и минимизации рисков для жизни экипажа.

Добавлено примечание ([EDG6]): Источник?

5. Формулировка задачи и цели разработки

5.1. Обоснование разработки

Эргономика играет важнейшую роль в обеспечении безопасности и эффективности работы пилотов. Анализ аварийных ситуаций в авиации показывает, что до 70% инцидентов связаны с человеческим фактором, включая усталость и перегрузку. [1]. Это делает создание системы для эргономической оценки кабины самолета актуальной задачей.

Добавлено примечание ([EDG7]): Источники???

Современные инструменты для оценки состояния операторов, такие как тесты PVT и NASA-TLX, применяются в различных отраслях, включая авиацию, медицину и военную сферу. Однако их интеграция в специализированное программное обеспечение для авиации остается ограниченной. Существующие решения, как правило, либо не адаптированы для условий авиационной среды, либо недостаточно удобны для использования в реальном времени. Это создает необходимость разработки новой системы, которая:

- Учитывает специфику работы пилотов.
- Интегрирует проверенные методики PVT и NASA-TLX.
- Позволяет объективно оценивать эргономические характеристики кабины самолета.

Разработка такой системы способствует повышению уровня безопасности полетов и снижению риска ошибок, связанных с человеческим фактором.

5.2. Цель и задачи разработки

5.2.1. Цель разработки

Создание минимально жизнеспособного продукта (MVP) программного обеспечения для эргономической оценки кабины самолета, использующего методики PVT и NASA-TLX для анализа когнитивной и физической нагрузки на пилотов.

5.2.2. Задачи разработки

1. **Сформировать требования к системе:**
    - Сбор информации о существующих решениях и методиках.
    - Определение функциональных и нефункциональных требований.
2. **Выбор стека технологий для реализации ПО:**
    - Анализ доступных библиотек и инструментов.
    - Обоснование выбора наиболее подходящей стека технологий.
3. **Разработать ключевые модули системы:**
    - Модуль для проведения теста PVT с реализацией визуализации стимулов и регистрации времени реакции.
    - Модуль NASA-TLX для ввода данных, расчета индекса нагрузки и визуализации.
4. **Создать графический пользовательский интерфейс:**
    - Проектирование интуитивно понятного GUI для тестирования и анализа данных.
    - Интеграция визуализации результатов с основными функциями системы.
5. **Обеспечить сохранение и управление данными:**
    - Реализация механизма хранения результатов тестов.
    - Добавление функций экспорта данных в удобный формат.
6. **Провести тестирование и валидацию системы:**
    - Проверка работоспособности всех модулей.
    - Проверка корректности отчетов

6. Требования

6.1. Условия использования системы

Система предназначена для проведения автономного анализа эргономических характеристик кабин самолета с использованием тестов PVT (Psychomotor Vigilance Test) и NASA-TLX (Task Load Index). Она должна работать в условиях ограниченного доступа к сети Интернет, чтобы обеспечить безопасность данных, так как летные данные представляют собой ценную информацию. Использование системы предполагает её установку на локальные машины, что исключает необходимость облачных сервисов и минимизирует риски утечки информации.

Для взаимодействия с системой предполагается использование стандартных устройств ввода информации (клавиатура и мышь), а также возможность просмотра и сохранения данных через интуитивно понятный графический интерфейс. Автономность системы является ключевым требованием, чтобы избежать зависимостей от внешних сервисов и инфраструктуры.

01_VC	Система должна функционировать в полном автономном режиме без необходимости подключения к сети Интернет.
02_VC	Система должна начинать процесс сбора данных и проведения тестирования в момент запуска через пользовательский интерфейс.
03_VC	Система должна работать с такими устройствами ввода информации, как клавиатура и мышь
04_VC	Система должна проводить анализ данных в соответствии с методиками тестов PVT и NASA-TLX, реализуя предопределенные алгоритмы обработки.
05_VC	Система должна блокировать возможность начала тестирования, если пользователь не прошел процесс

	регистрации или авторизации через предоставленный интерфейс.
06_VC	Система должна предоставлять пользователю визуальное представление результатов тестов в графическом интерфейсе.
07_VC	Система должна сохранять результаты каждого теста для последующего анализа и экспорта.
08_VC	Система должна использовать компоненты с открытым исходным кодом.
09_VC	Система должна минимизировать требования к ресурсам, чтобы работать на устройствах с ограниченными вычислительными возможностями.

7. Выбор стека технологий

7.2 Анализ существующих решений

На текущий момент не существует интегрированных программных продуктов, специально предназначенных для оценки эргономики, которые одновременно включают тесты PVT (Psychomotor Vigilance Task) и NASA-TLX (NASA Task Load Index). Однако, отдельные инструменты для проведения каждого из этих тестов доступны и широко используются в различных исследованиях и практиках.

7.1.1. Инструменты для проведения теста PVT:

PVT-192 — портативное устройство, предназначенное для измерения времени реакции в лабораторных и полевых условиях. Оно используется для оценки уровня бдительности и выявления когнитивных нарушений, связанных с усталостью. Принцип действия PVT-192 основан на измерении времени реакции пользователя на визуальные стимулы, что позволяет объективно оценить психомоторную скорость и внимание. Устройство оснащено внутренним аккумулятором и может работать автономно, что делает его удобным для длительных исследований в различных условиях.

Рис. 7.1. Устройство PVT-192

PVT Self-Test — мобильное приложение, доступное для различных платформ, которое предоставляет возможность самостоятельно проводить тест PVT. Оно используется в исследованиях сна и бдительности, а также для индивидуального мониторинга уровня усталости. Принцип действия приложения аналогичен устройству PVT-192: пользователь реагирует на визуальные стимулы, а приложение фиксирует время реакции, позволяя оценить когнитивное состояние.

Оба инструмента широко применяются в научных исследованиях, связанных с изучением влияния усталости на когнитивные функции, а также в практических областях, требующих мониторинга уровня бдительности операторов, таких как авиация, транспорт и медицина

7.1.2. Инструменты для проведения NASA-TLX:

Первоначальная версия инструмента, известная как NASA-TLX Paper-and-Pencil Version, представляет собой бумажный опросник. Участник после выполнения задания оценивает шесть аспектов нагрузки по шкале от 0 до 100. Затем происходит взвешивание значимости этих аспектов путём парных сравнений, что позволяет учесть особенности восприятия конкретной задачи. Итоговый индекс нагрузки рассчитывается с учётом весовых коэффициентов, предоставляя точную количественную оценку рабочей нагрузки. Эта версия проста в использовании и до сих пор применяется в ситуациях, где цифровые инструменты недоступны или нецелесообразны.

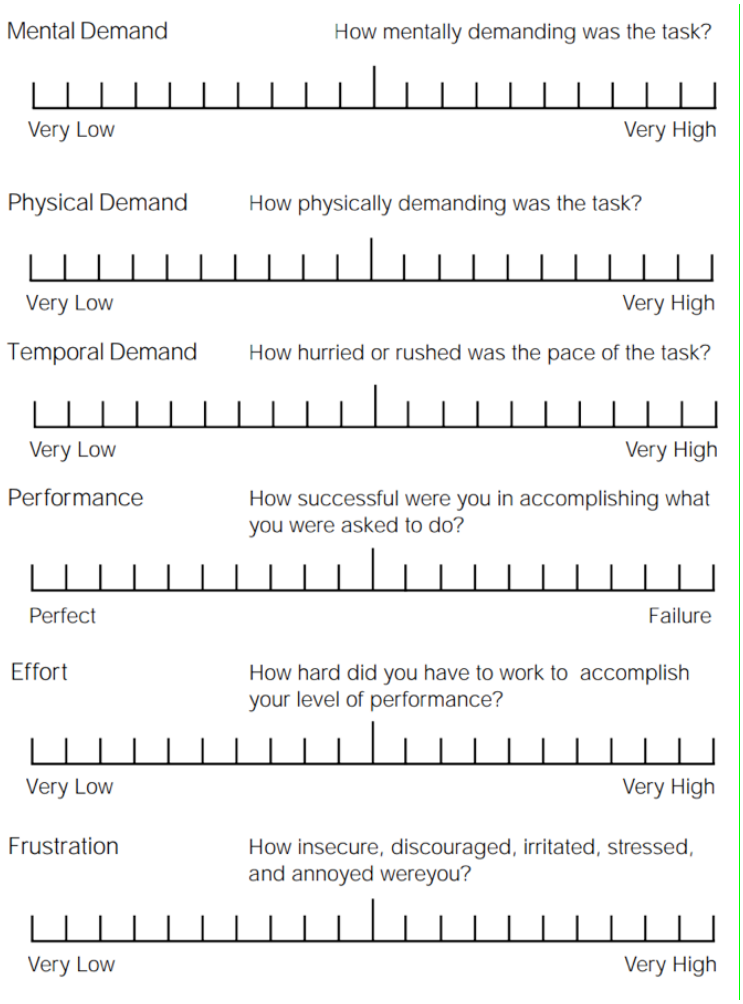
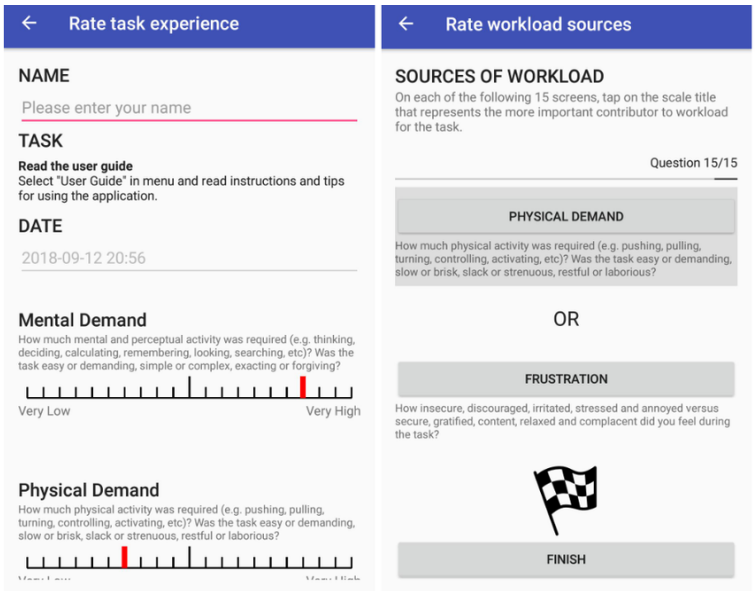


Рис 7.2. Бумажной версии NASA-TLX

С развитием технологий NASA представила цифровую версию инструмента — NASA-TLX App. Это приложение предоставляет все возможности бумажной версии, но дополнительно упрощает процесс сбора и анализа данных. Удобный интерфейс позволяет участникам быстро заполнять оценки, а исследователи получают мгновенный доступ к обработанным результатам. Возможности приложения делают его незаменимым инструментом для современных исследований, где требуется

оперативный анализ данных. NASA-TLX App активно используется в тестировании интерфейсов, авиационных систем и медицинских приборов.

Рис 7.2. Электронная версии NASA-TLX (приложение от компании NASA)



Применение NASA-TLX выходит далеко за пределы лабораторий. В авиации инструмент позволяет анализировать нагрузку на пилотов при выполнении сложных манёвров или в стрессовых условиях, таких как посадка при плохой видимости. В медицинской сфере он помогает оценивать рабочую нагрузку врачей и медсестёр в интенсивных условиях, например, в операционных или отделениях интенсивной терапии. В области разработки пользовательских интерфейсов NASA-TLX используется для оптимизации дизайна, чтобы минимизировать когнитивную нагрузку пользователей.

Интересно, что оба формата инструмента, бумажный и цифровой, имеют свои преимущества. Бумажная версия остаётся востребованной благодаря своей доступности и независимости от технической инфраструктуры, что особенно важно в полевых условиях или при ограниченных ресурсах. Цифровая версия, напротив, идеально подходит для

масштабных исследований, где требуется автоматизация сбора данных и их анализ в реальном времени.

7.1.3. Результаты анализа существующих решений:

Несмотря на наличие отдельных инструментов для проведения тестов PVT и NASA-TLX, а также комплексных систем для общей эргономической оценки, на рынке отсутствует интегрированное решение, объединяющее обе методики в едином программном продукте. Это подчеркивает актуальность разработки подобного приложения, которое могло бы предоставить возможность проведения оценки эргономики с использованием нескольких методик в едином интерфейсе.

7.2 Анализ подходящих технологий

При разработке программного обеспечения важным этапом является выбор библиотек и инструментов, которые обеспечат необходимую функциональность, стабильность и удобство использования. Для проекта, ориентированного на оценку эргономики кабины самолета, особое внимание уделено таким критериям, как открытый исходный код, кроссплатформенность, поддержка автономного режима работы и переносимость данных между устройствами.

Для реализации проекта выбран язык программирования Python версии 3.10. Python предоставляет мощный и гибкий инструмент для разработки благодаря своей читаемости, богатой экосистеме библиотек и поддержке кроссплатформенности.

Версия 3.10 была выбрана из-за её стабильности и набора возможностей, таких как улучшенная поддержка типов и сопоставление с образцом (pattern matching), которые облегчают разработку сложных приложений.

Python также широко используется в научных исследованиях благодаря своей простоте и обширному набору библиотек, таких как NumPy, SciPy и Matplotlib. Эти библиотеки предоставляют мощные инструменты для математического и статистического анализа, что делает Python подходящим

выбором для создания приложения, ориентированного на эргономический анализ. Его кроссплатформенность гарантирует, что разработанное приложение будет работать одинаково эффективно на Windows, macOS и Linux.

Графический интерфейс приложения построен с использованием библиотеки Tkinter. Это стандартная библиотека Python, которая имеет открытый исходный код и является частью стандартной поставки языка. Tkinter обеспечивает простоту разработки и интеграции, что делает её подходящей для минимально жизнеспособного продукта (MVP). В отличие от альтернатив, таких как PyQt или Kivy, она не требует сложной настройки и лицензирования. Например, PyQt обладает расширенным функционалом, но его использование ограничивается лицензией GPL, что делает его менее подходящим для опенсорсных проектов. Kivy, хотя и является кроссплатформенной библиотекой, больше ориентирован на мобильные приложения, что ограничивает её применение для настольных решений.

Таблица 7.2.1 Характеристики библиотек для пользовательских интерфейсов

Библиотека	**Открытый исходный код**	**Кроссплатформенность**	**Простота интеграции**	**Лицензия**
Tkinter	Да	Да	Высокая	MIT
PyQt	Да	Да	Средняя	GPL
Kivy	Да	Да	Средняя	MIT

Для хранения данных была выбрана база данных SQLite. Её основные преимущества — лёгкость, кроссплатформенность и переносимость. В отличие от серверных решений, таких как PostgreSQL или MySQL, SQLite не требует настройки серверного окружения. Все данные хранятся в одном файле, что упрощает перенос приложения между устройствами. Этот выбор особенно актуален для автономной работы приложения, когда необходимо обеспечить доступ к данным без подключения к серверу.

Таблица 7.2.2 Характеристики баз данных

База данных	**Открытый исходный код**	**Кроссплат форменность**	**Переносимость данных**	**Простота использования**
SQLite	Да	Да	Высокая	Высокая
PostgreSQL	Да	Да	Низкая	Средняя
MySQL	Да	Да	Низкая	Средняя

Для создания исполняемого файла выбрана библиотека PyInstaller. Она позволяет упаковать Python-приложение в один файл, что значительно упрощает его использование и распространение. PyInstaller поддерживает работу на Windows, macOS и Linux, что делает её универсальным инструментом для сборки. В отличие от cx_Freeze и py2exe, PyInstaller требует меньше усилий для настройки и предоставляет более широкий спектр возможностей.

Таблица 7.2.3 Характеристики библиотек для создания исполняемого файла

Инструмент	**Открытый исходный код**	**Кроссплатформенность**	**Простота использования**
PyInstaller	Да	Да	Высокая
cx_Freeze	Да	Да	Средняя
py2exe	Да	Нет	Средняя

Для взаимодействия с базой данных используется SQLAlchemy. Эта библиотека сочетает в себе гибкость и мощность, позволяя работать как на уровне ORM, так и с чистыми SQL-запросами. Её конкурентами являются Django ORM и Peewee, однако Django ORM требует использование всего фреймворка Django, что избыточно для данного проекта. Peewee, хотя и является легковесным, уступает SQLAlchemy в функциональности и поддержке.

Таблица 7.2.4 Характеристики библиотек для взаимодействия с базой данных

ORM	Открытый исходный код	Гибкость	Интеграция с SQLite
SQLAlchemy	Да	Высокая	Да
Django ORM	Да	Средняя	Да
Peewee	Да	Низкая	Да

Для работы с данными внутри приложения используется Pydantic. Эта библиотека обеспечивает валидность данных и их строгое соответствие типам Python. Она выбрана благодаря простоте использования, скорости и поддержке современных подходов к работе с данными. Альтернативы, такие как Cerberus и Marshmallow, предоставляют схожие функции, но уступают Pydantic в производительности и удобстве.

Таблица 7.2.5 Характеристики библиотек для взаимодействия с базой данных

Библиотека	Открытый исходный код	Простота использования	Производительность
Pydantic	Да	Высокая	Высокая
Cerberus	Да	Средняя	Средняя
Marshmallow	Да	Средняя	Средняя

Для обработки данных, их анализа и визуализации в проекте используются библиотеки NumPy, Pandas и Matplotlib. Эти библиотеки обеспечивают удобство работы с большими объёмами данных, высокую производительность вычислений и широкие возможности для представления информации. Для выбора именно этих библиотек проведено сравнение с их аналогами.

NumPy — ключевой инструмент для работы с многомерными массивами и матрицами. Его производительность значительно превосходит стандартные возможности Python за счёт использования низкоуровневых оптимизаций. Конкурентами **NumPy** являются **SciPy** и **TensorFlow**, которые предоставляют схожий функционал, но ориентированы на более узкие области применения. SciPy используется для научных расчётов, а TensorFlow

— для машинного обучения, что делает NumPy наиболее универсальным выбором.

Таблица 7.2.6 Характеристики библиотек для вычислений и визуализации

Библиотека	**Открытый исходный код**	**Простота использования**	**Производительность**	**Специализация**
NumPy	Да	Высокая	Высокая	Общие вычисления
SciPy	Да	Средняя	Высокая	Научные расчёты
TensorFlow	Да	Низкая	Высокая	Машинное обучение

Pandas — лидер в обработке табличных данных и временных рядов. Конкурентами являются **Dask** и **Vaex**, которые обеспечивают параллельную обработку больших данных. Однако Pandas остаётся лучшим выбором для небольших и средних проектов из-за простоты использования и доступности инструментов для работы с данными.

Таблица 7.2.7 Характеристики библиотек для обработки табличных данных

Библиотека	**Открытый исходный код**	**Простота использования**	**Поддержка больших данных**
Pandas	Да	Высокая	Средняя
Dask	Да	Средняя	Высокая
Vaex	Да	Средняя	Высокая

Matplotlib — стандарт для создания графиков и визуализации данных. Конкуренты, такие как **Seaborn** и **Plotly**, предлагают более современные интерфейсы и дополнительные функции. Однако Matplotlib обеспечивает высокий уровень контроля над элементами графиков и полностью поддерживает работу в автономном режиме, что делает её оптимальной для проектов без подключения к сети.

Таблица 7.2.8 Характеристики библиотек для обработки табличных данных

Библиотека	**Открытый исходный код**	**Простота использования**	**Гибкость настройки**	**Автономность**
Matplotlib	Да	Высокая	Высокая	Да
Seaborn	Да	Высокая	Средняя	Да
Plotly	Да	Средняя	Высокая	Нет

Выбор NumPy, Pandas и Matplotlib продиктован их универсальностью, производительностью и открытым исходным кодом. Эти библиотеки идеально подходят для обработки и анализа данных в рамках автономного приложения, обеспечивая необходимые инструменты для реализации поставленных задач.

8. Определения технического задания

Разработка программного обеспечения для оценки эргономики кабины самолета начинается с анализа исходных данных и условий, в которых оно будет использоваться. Главной целью является создание инструмента, который способен проводить тестирование операторов с применением методик PVT и NASA-TLX для оценки когнитивной нагрузки и уровня бдительности. Важно, чтобы это программное обеспечение соответствовало специфическим требованиям, включая автономность, простоту использования и способность работать в кроссплатформенном режиме.

Программное обеспечение должно быть полностью автономным, исключая необходимость подключения к сети во время работы. Это обусловлено возможностью эксплуатации в условиях ограниченного доступа к сети. Также важно, чтобы система поддерживала корректное функционирование на наиболее популярных операционных системах, таких как Windows, Linux, что делает её более универсальной. Среда эксплуатации предполагает использование персональных компьютеров или ноутбуков с минимальными техническими характеристиками, такими как процессор с тактовой частотой 2 GHz, оперативная память объёмом 4 GB и свободное место на диске в 200 MB.

Основными пользователями программного обеспечения станут пилоты, которые проходят тестирование для определения их когнитивной нагрузки, а также исследователи и инженеры, занимающиеся анализом полученных данных. Для этих категорий важно, чтобы приложение было интуитивно понятным, легко настраиваемым и обеспечивало визуализацию результатов в удобной форме. Кроме того, оно должно предоставлять возможность сохранения результатов в локальной базе данных для последующего анализа и экспортировать их в формате CSV или JSON, что позволит интегрировать данные с другими системами или включать их в отчёты.

Программное обеспечение должно учитывать ряд технических ограничений. Например, оно должно работать в среде с ограниченным

доступом к внешним ресурсам и быть оптимизированным для минимальных системных требований. Функциональные возможности включают проведение тестов PVT и NASA-TLX, расчёт ключевых метрик, таких как средняя скорость реакции и уровень рабочей нагрузки, а также визуализацию данных в виде графиков и таблиц.

Простота установки и настройки является ключевым нефункциональным требованием. Пользовательский интерфейс должен быть разработан с использованием библиотеки Tkinter, что обеспечит интуитивно понятное взаимодействие даже для пользователей с минимальными техническими знаниями. Производительность приложения должна быть на высоком уровне, чтобы минимизировать нагрузку на систему и обеспечить стабильную работу даже на компьютерах с базовыми характеристиками.

На основании вышеуказанных условий сформировано техническое задание. Программное обеспечение должно быть создано на языке Python версии 3.10 с использованием библиотек с открытым исходным кодом, таких как Tkinter для интерфейса, SQLite для работы с базой данных, а также SQLAlchemy, NumPy, Pandas и Matplotlib для обработки и визуализации данных. Основной функционал включает реализацию тестов PVT и NASA-TLX, сохранение и экспорт данных. Программа должна быть полностью автономной и оптимизированной для работы на системах с минимальными характеристиками.

9. Разработка программного обеспечения

Для реализации логики программы выбрана интегрированная среда разработки (IDE) **PyCharm Community Edition**, которая идеально подходит для разработки на языке Python благодаря своей удобной функциональности, подсветке синтаксиса, инструментам отладки и интеграции с системами управления версиями, такими как Git.

Разработка велась на платформе операционной системы Windows, которая обеспечивала стабильность работы и совместимость с инструментами, необходимыми для создания приложения.

В рамках проекта создано приложение с рабочим названием **"Ergonomics_Assessment"**, которое было реализовано с использованием таких библиотек, как Tkinter для разработки пользовательского интерфейса, SQLAlchemy для управления базой данных SQLite, а также Pydantic и NumPy для обработки данных.

При сборке проекта используется PyInstaller, который позволяет упаковать приложение в единый исполняемый файл. Это обеспечивает удобство распространения и возможность использования приложения на различных системах без необходимости установки дополнительных зависимостей.

Итогом сборки является файл **"Ergonomics_Assessment.exe"**, который включает весь необходимый функционал для выполнения тестов, обработки данных и предоставления результатов.

9.1. Архитектура приложения

Архитектура приложения была спроектирована таким образом, чтобы обеспечить её гибкость, модульность и удобство в поддержке.

При разработке был выбран модульный подход, который позволяет чётко разделить ответственность между компонентами, упрощая их тестирование, модификацию и масштабирование.

Основой для взаимодействия модулей стало использование современных паттернов проектирования, таких как "Фасад" и "Репозиторий", которые

помогают упорядочить структуру приложения и сделать её интуитивно понятной для разработчиков.

Архитектура включает три основных слоя: пользовательский интерфейс, сервисный слой и слой данных. Каждый из них выполняет свою чётко определённую задачу, обеспечивая надежную и стабильную работу системы.

9.1.1. Слой пользовательского интерфейса (UI)

Пользовательский интерфейс отвечает за взаимодействие с конечным пользователем. Для его реализации была выбрана библиотека Tkinter, которая предоставляет инструменты для создания простых, но функциональных графических интерфейсов.

Пользовательский интерфейс включает несколько экранов (фреймов), каждый из которых отвечает за выполнение своей задачи: авторизация, регистрация, проведение тестов и отображение результатов.

Управление экранами осуществляется через специальный компонент — менеджер окон (WindowManager). Этот менеджер обеспечивает переключение между экранами, передачу данных и их обновление.

Например, если пользователь завершает тест PVT, результаты этого теста автоматически передаются на экран завершения, где пользователь может ознакомиться с итогами.

Менеджер окон был реализован с применением паттерна "Синглтон", что гарантирует существование единственного экземпляра класса на протяжении всего времени работы приложения. Это исключает вероятность создания дублирующих менеджеров и обеспечивает централизованное управление окнами. Пример реализации класса:

```
# ui/window_manager.py

class WindowManager:
    _instance = None

    def __new__(cls, *args, **kwargs):
        if cls._instance is None:
            cls._instance = super(WindowManager, cls).__new__(cls)
        return cls._instance
```

```
    def __init__(self, root):
        self.root = root
        self.frames = {}
        self.current_user_id = None  # <-- здесь храним ID пользователя
после логина

        self.root.grid_rowconfigure(0, weight=1)
        self.root.grid_columnconfigure(0, weight=1)

    def create_and_register(self, name, frame_class, *args, **kwargs):
        frame = frame_class(self.root, *args, **kwargs)
        frame.grid(row=0, column=0, sticky="nsew")
        self.frames[name] = frame

    def show_frame(self, name, **kwargs):
        frame = self.frames.get(name)
        if frame:
            if hasattr(frame, "update_data"):
                frame.update_data(**kwargs)
            frame.tkraise()
        else:
            raise ValueError(f"Frame '{name}' не найден!")
```

Использование паттерна "Синглтон" для менеджера окон позволяет сократить количество создаваемых объектов и улучшить управление ресурсами приложения.

9.1.2. Сервисный слой

Сервисный слой реализует бизнес-логику приложения. Он обрабатывает действия пользователя, взаимодействует с базой данных через слой данных и предоставляет результаты для отображения в интерфейсе.

Для упрощения работы и структурирования кода в сервисном слое используются паттерны "Фасад" и "Репозиторий".

Паттерн "Фасад" позволяет организовать доступ к функционалу через единый интерфейс, скрывая сложную внутреннюю реализацию. Например, сервис работы с пользователями (UserService) включает методы для проверки логина, создания нового пользователя, обновления данных и удаления записей.

Это делает работу с пользователями более упорядоченной и минимизирует количество кода, необходимого для выполнения операций.

Пример кода из сервиса работы с пользователями:

```
class UserService:
```

```
    def is_login_exists(self, login: str) -> bool:
        with UserRepository() as repo:
            return repo.is_login_exists(login)

    def create_user(self, user_data: dict) -> int:
        with UserRepository() as repo:
            return repo.create_user(user_data)

    def get_user_by_id(self, user_id: int) -> Optional[User]:
        with UserRepository() as repo:
            return repo.get_user_by_id(user_id)

    def check_user(self, login: str, password: str) -> tuple[str, int] |
None:
        """
        Возвращает (FIO, user_id), если логин и пароль правильные,
        иначе None.
        """
        with UserRepository() as repo:
            result = repo.check_user_with_id(login, password)
            if result:
                return result
            return None

    def update_user(self, user_id: int, update_data: dict) -> User:
        with UserRepository() as repo:
            return repo.update_user(user_id, update_data)

    def delete_user(self, user_id: int) -> None:
        with UserRepository() as repo:
            repo.delete_user(user_id)

    def get_all_users(self) -> list[Type[User]]:
        with UserRepository() as repo:
            return repo.get_all_users()
```

Пример кода из сервиса работы с тестами:

```
class TestService:

    def save_pvt_round(
        self,
        user_id: int,
        exercise_name: str,
        task_number: str,
        type_test: str,
        round_index: int,
        reaction_time: float
    ) -> PVTResult:
        with TestRepository() as repo:
            return repo.save_pvt_round(
                user_id=user_id,
                exercise_name=exercise_name,
                task_number=task_number,
                type_test=type_test,
                round_index=round_index,
                reaction_time=reaction_time
            )

    def get_pvt_results_for_user(self, user_id: int) -> List[PVTResult]:
        with TestRepository() as repo:
            return repo.get_pvt_results_for_user(user_id)

    def save_nasa_tlx_result(
```

```
        self,
        user_id: int,
        exercise_name: str,
        task_number: str,
        mental_demand: int,
        physical_demand: int,
        temporal_demand: int,
        performance: int,
        effort: int,
        frustration: int,
        weight_mental: int,
        weight_physical: int,
        weight_temporal: int,
        weight_performance: int,
        weight_effort: int,
        weight_frustration: int,
        weighted_tlx: float
    ) -> NASATLXResult:
        with TestRepository() as repo:
            return repo.save_nasa_tlx_result(
                user_id=user_id,
                exercise_name=exercise_name,
                task_number=task_number,
                mental_demand=mental_demand,
                physical_demand=physical_demand,
                temporal_demand=temporal_demand,
                performance=performance,
                effort=effort,
                frustration=frustration,
                weight_mental=weight_mental,
                weight_physical=weight_physical,
                weight_temporal=weight_temporal,
                weight_performance=weight_performance,
                weight_effort=weight_effort,
                weight_frustration=weight_frustration,
                weighted_tlx=weighted_tlx
            )

    def get_nasa_tlx_results_for_user(self, user_id: int) ->
List[NASATLXResult]:
        with TestRepository() as repo:
            return repo.get_nasa_tlx_results_for_user(user_id)
```

9.1.3. Слой данных

Слой данных отвечает за взаимодействие с базой данных, которая хранит всю необходимую информацию о пользователях и результатах тестов. База данных построена на основе SQLite, что делает приложение лёгким и кроссплатформенным.

Для работы с базой данных используется SQLAlchemy, предоставляющая ORM для управления данными через объекты Python.

Важной особенностью является использование паттерна "Репозиторий" для всех операций с базой данных. Это гарантирует строгое разделение между бизнес-логикой и данными, что повышает безопасность и снижает

вероятность ошибок. Репозиторий инкапсулирует все операции, включая создание, чтение, обновление и удаление записей, а также выполняет валидацию данных через Pydantic.

Пример реализации репозитория для работы с пользователями:

```
class UserRepository:
    def __init__(self):
        self.db: Session = None

    def __enter__(self):
        self.db = SessionLocal()
        return self

    def __exit__(self, exc_type, exc_value, traceback):
        if self.db:
            self.db.close()

    def is_login_exists(self, login: str) -> bool:
        return self.db.query(User).filter(User.login == login).first() is
not None

    def create_user(self, user_data: dict) -> int:
        try:
            validated_data = UserSchema(**user_data)
        except ValidationError as e:
            error_details = [
                {"field": err["loc"][0], "message": err["msg"]}
                for err in e.errors()
            ]
            raise ValueError(f"Ошибка валидации: {error_details}")

        if self.is_login_exists(validated_data.login):
            raise ValueError("Пользователь с таким логином уже существует!")

        user = User(**validated_data.dict())
        self.db.add(user)
        self.db.commit()
        self.db.refresh(user)
        return user.id

    def get_user_by_id(self, user_id: int) -> Optional[User]:
        return self.db.query(User).filter(User.id == user_id).first()

    def check_user(self, login: str, password: str) -> str | None:
        return self.db.query(User.name).filter(User.login == login,
User.password == password).scalar()
```

Пример реализации репозитория для работы с тестами:

```
class TestRepository:
    def __init__(self):
        self.db: Session = None

    def __enter__(self):
        self.db = SessionLocal()
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        if self.db:
```

```
            self.db.close()

    # ---------- PVT ----------

    def save_pvt_round(
        self,
        user_id: int,
        exercise_name: str,
        task_number: str,
        type_test: str,
        round_index: int,
        reaction_time: float
    ) -> PVTResult:
        """
        Сохраняет один раунд PVT-теста.
        Возвращает объект PVTResult.
        """
        result = PVTResult(
            user_id=user_id,
            exercise_name=exercise_name,
            task_number=task_number,
            type_test=type_test,
            round_index=round_index,
            reaction_time=reaction_time
        )
        self.db.add(result)
        self.db.commit()
        self.db.refresh(result)
        return result

    def get_pvt_results_for_user(self, user_id: int) -> List[PVTResult]:
        return (
            self.db.query(PVTResult)
            .filter(PVTResult.user_id == user_id)
            .order_by(PVTResult.timestamp.desc())
            .all()
        )

    # ---------- NASA-TLX ----------

    def save_nasa_tlx_result(
        self,
        user_id: int,
        exercise_name: str,
        task_number: str,
        mental_demand: int,
        physical_demand: int,
        temporal_demand: int,
        performance: int,
        effort: int,
        frustration: int,
        weight_mental: int,
        weight_physical: int,
        weight_temporal: int,
        weight_performance: int,
        weight_effort: int,
        weight_frustration: int,
        weighted_tlx: float
    ) -> NASATLXResult:
        """
        Сохраняет итог NASA-TLX вместе с сырыми баллами и весами.
        """
        result = NASATLXResult(
```

```
            user_id=user_id,
            exercise_name=exercise_name,
            task_number=task_number,
            mental_demand=mental_demand,
            physical_demand=physical_demand,
            temporal_demand=temporal_demand,
            performance=performance,
            effort=effort,
            frustration=frustration,
            weight_mental=weight_mental,
            weight_physical=weight_physical,
            weight_temporal=weight_temporal,
            weight_performance=weight_performance,
            weight_effort=weight_effort,
            weight_frustration=weight_frustration,
            weighted_tlx=weighted_tlx
        )
        self.db.add(result)
        self.db.commit()
        self.db.refresh(result)
        return result
```

9.1.4. Преимущества архитектуры

Основная цель архитектуры — обеспечить простоту в разработке, удобство для пользователя и стабильность системы. Среди основных преимуществ:

- **Модульность:** Каждый компонент отвечает за свою часть функционала, что облегчает тестирование и добавление новых возможностей.
- **Гибкость:** Паттерны "Фасад" и "Репозиторий" позволяют легко изменять внутреннюю реализацию без влияния на интерфейсы.
- **Читаемость:** Чёткое разделение слоёв делает структуру кода интуитивно понятной.
- **Кроссплатформенность:** Использование Tkinter и SQLite позволяет запускать приложение на Windows и Linux.

Эта архитектура не только отвечает всем требованиям к функциональности, но и создаёт основу для дальнейшего развития приложения.

9.1.5. Диаграмма архитектуры

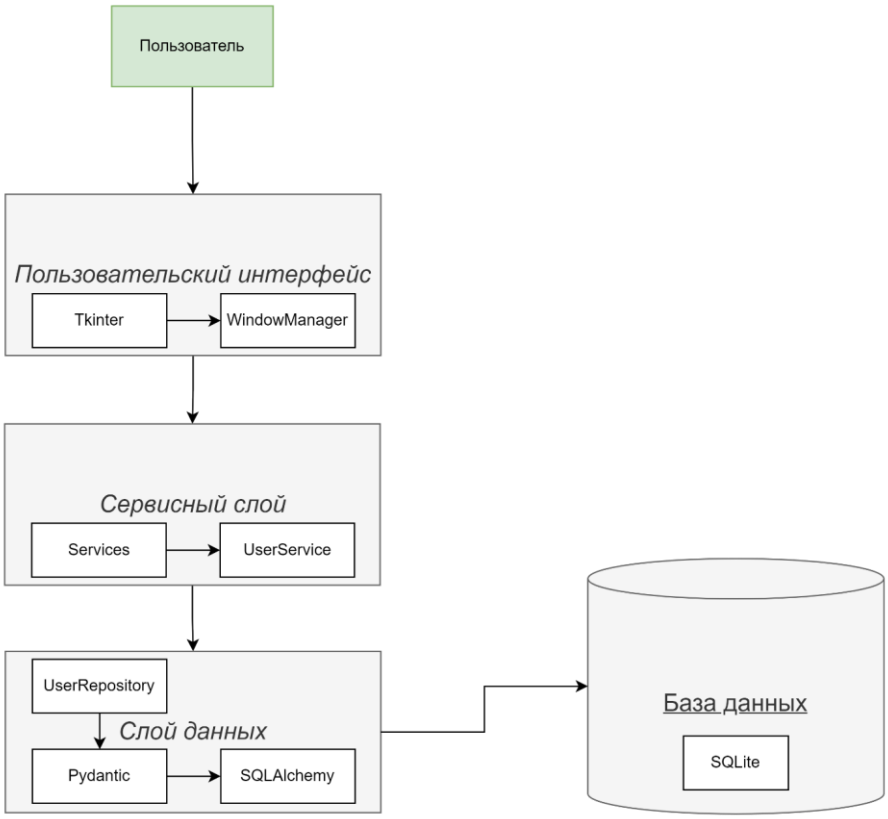


Рис. 9.1.5.1 3.1.1. Диаграмма архитектуры

10.Структура проекта

Структура проекта организована в виде нескольких ключевых модулей и папок, каждая из которых выполняет определённые функции. Проект состоит из следующих директорий и файлов:

```
ErgonomicsApplication
|   app.py
|   config.py
|   README.md
|   requirements.txt
|
+---app
|   |   __init__.py
|   |
|   +---database
|   |       database.py
|   |       models.py
|   |       __init__.py
|   |
|   +---repository
|   |       TestRepository.py
|   |       UserRepository.py
|   |       __init__.py
|   |
|   +---schemas
|   |       TestResultsSchema.py
|   |       UserSchema.py
|   |       __init__.py
|   |
|   \---services
|           TestService.py
|           UserService.py
|           __init__.py
|
+---tests
|       test_nasa_tlx_logic.py
|       test_pvt_logic.py
|       test_user_service.py
|       __init__.py
|
\---ui
    |   window_manager.py
    |   __init__.py
    |
    +---assets
    |   |   logo.png
    |   |
    |   \---icons
    |           test.png
    |           user.png
    |
    +---components
    |       CenteredFrame.py
    |       ErrorLabel.py
    |       StyledButton.py
    |       __init__.py
    |
    \---screens
            AuthWindow.py
            LoginWindow.py
```

```
            MainWindow.py
            NASA_TLXWindow.py
            PVTWindow.py
            RegisterWindow.py
            ReportWindow.py
            ResultsWindow.py
            TestInstructionWindow.py
            TestSelectionWindow.py
            __init__.py
```

app — включает в себя основную бизнес-логику, модели данных и взаимодействие с базой данных:

- **schemas**: Содержит схемы для валидации данных с использованием библиотеки Pydantic.
    - UserSchema.py: Определяет структуру данных для пользователя.
    - TestResultsSchema.py: Определяет структуру результатов тестирования.
- **services**: Реализует слой бизнес-логики. Например, UserService.py отвечает за управление данными пользователей.
- **repository**: Содержит репозитории для работы с базой данных. Репозитории инкапсулируют операции CRUD.
- **database**: Включает модели базы данных и функции для инициализации подключения.

ui — отвечает за графический интерфейс приложения:

- **screens**: Содержит файлы, представляющие основные экраны приложения, такие как авторизация (AuthWindow.py), выбор теста (TestSelectionWindow.py), тесты (PVTWindow.py, NASA_TLXWindow.py) и результаты (ResultsWindow.py).
- **components**: Содержит переиспользуемые элементы интерфейса, например кнопки с определённым стилем (StyledButton.py) и центрированные фреймы (CenteredFrame.py).
- **assets**: Хранит статические ресурсы, такие как изображения и иконки.

tests — модуль для автоматического тестирования. Включает тесты для проверки работы бизнес-логики, таких как логика проведения тестов PVT и NASA-TLX.

Заключение

1. Основные выводы работы.
2. Перспективы дальнейших исследований.

Список использованных источников

(формируется согласно ГОСТ).

Приложения

1. Примеры интерфейса приложения.
2. Исходные данные для тестирования.
3. Техническое описание алгоритмов.ы