Система учета рейтинга успеваемости студентов “Рейтинг-калькулятор”

Комментарий к документу. Образцы документов по оформлению лабораторных и контрольных работ по дисциплине “Программная инженерия” для различных тех.процессов жизненного цикла ПО (ТП ЖЦ)

Видение проекта

Комментарий к документу. Предварительное обоснование проекта, включающее в себя:

- название проекта
- цели проекта
- результаты проекта
- допущения и ограничения
- ключевые участники и заинтересованные стороны
- ресурсы проекта
- сроки
- риски
- критерии приемки
- обоснование полезности проекта

Для некоммерческих, исследовательских и инновационных проектов возможна более свободная форма обоснования, включающая в себя:

- *бизнес-требования* – обоснование полезности проекта, особенности проекта, обеспечивающие его привлекательность,предполагаемые отличия от аналогов, проблемы предметной области и способы их решения в проекте, возможности коммерческого использования,способы монетизации.
- *границы проекта* – перечень бизнес-процессов, поддерживаемых и не поддерживаемых системой.
- *перечень пользователей* проекта

Бизнес-требования

Вот смотри, - сказал он. - Эта самая маленькая монетка называется сантик, а вот эта, побольше, - два сантика; вот еще такая же монетка тоже два сантика, вот еще две монеты по пять сантиков, видишь? Всего, значит, у меня пятнадцать сантиков. А сто сантиков составляют один фертинг.

Н.Носов «Незнайка на Луне»

Актуальность

При достаточно высоком уровне информатизации делопроизводства и учебного процесса в НГТУ (учебные планы, нагрузка преподавателей, успеваемость в сессию, приказы) контроль текущей успеваемости ведется по старинке. В то же время введение балльно-рейтинговой системы со 100-балльной шкалой требует фиксации большого объема данных о сроках и качестве выполненных работ. Общие положения о вычислении рейтинга определены в нормативных

документах НГТУ, конкретные данные (единицы контроля, нормативные баллы, правила вычисления рейтинга) должны быть определены в рабочих программах преподавателей.

Обоснование полезности проекта

Проект не направлен на получение коммерческой выгоды. Полезными эффектами проекта являются:

- повышение объективности оценок и оперативности контроля успеваемости
- открытость и доступность данных о текущей успеваемости
- возможность экспорта накопленных данных
- повышение имиджа учебного заведения

Основные проблемы, требующие решения

- приемлемая формализация процесса вычисления рейтинга
- очевидность оценки в стандартных ситуациях
- использование мобильных клиентских приложений в том числе и при недоступности БД (отсутствие или низкое качество сетевых соединений в аудиториях)
- ведение архива отчетных документов
- система авторизации с учетом доступа к персональным данным
- степень централизации и затраты на обслуживание

Формальная схема вычисления рейтинга

Общий рейтинг определяется как сумма рейтингов по единицам контроля минус снижение за пропуски занятий:

- **способы подсчета -** ручной (экспертный, субъективный)– исходный балл ставится преподавателем в пределах указанного максимума, система в формировании оценки участия не принимает (экзамен, зачет, индивидуальное задание, курсовая работа), автоматический **–** исходный балл устанавливается по нормативу и снижается/увеличивается по формальным критериям;
- **формальные критерии качества –** при автоматическом подсчете балла введен список критериев (например, небрежность оформления, повышенная сложность), по которым производится увеличение/снижение балла на фиксированный процент по каждому из них (например, на 10%). Признаки можно выбирать группой.
- **учет сроков сдачи** – для ряда учебных единиц (лабораторная работа, защита работы, контрольная работа) вводится линейная шкала снижения балла на заданный процент за каждую просроченную неделю относительно установленного срока сдачи. Срок сдачи каждой учебной единицы устанавливается при создании рейтинга в соответствии с расписанием (например, срок сдачи лабораторной работы – следующее занятие за ее выполнением). Число недель, в течение которых балл снижается, ограничено, т.е. вводится предельное снижение (например, 50%). При досрочной сдаче производится симметричное увеличение;
- **учет посещаемости** – для некоторых видов учебных единиц предусмотрен контроль посещаемости (лабораторные работы), они вносятся в отдельный список при создании рейтинга, этот список может дополняться преподавателем. Поскольку фиксируется факт пропуска занятия, за который в рейтинге предусмотрено фиксированное снижение рейтинга (например, 0.5 балла за пропуск), то учет посещаемости не является обязательным;
- **весовые коэффициенты вычисления рейтинга** составляют типовые наборы и назначаются отдельно каждому рейтингу:

- процент снижения (увеличения) балла по сроку сдачи (7-10% за неделю);
- количество недель, в течение которых действует предыдущее снижение (увеличение) – (5-7 недель, т.е. максимум 50% в обе стороны);
- процент изменения балла за каждый параметр качества (10%);
- количество баллов, снимаемых за каждый пропуск (0.5-1 балл).

Очевидность оценки в стандартных ситуациях

Очевидной выгодой предлагаемой схемы является естественное соотношение “закона и справедливости” в стандартных ситуациях, что делает процедуру оценки более прозрачной и снижает психологическую напряженность:

- *«Двоек не ставим».* Если вся учебная нагрузка выполнена в срок и без претензий (но и без «эксклюзива»), то студент зарабатывает рейтинг, достаточный для получения «тройки» на экзамене. При этом средний и высокий уровень ответов на экзамене оцениваются соответственно.
- *«Досрочно и автоматом».* Если студент сдает все досрочно, то ему достаточно небольшого индивидуального задания для получения «автомата». Можно также сдавать часть экзамена, например, без теории.
- *«Так себе, на троечку».* При наличии небольших долгов (невыполнение или сдача не в срок), студент может быть допущен к экзамену, но по общему рейтингу выше «четверки» не получит, а реально может претендовать только на «тройку»;
- *«Оптом не дешевле».* Если студент в течение семестра не занимается, то, сдав в конце семестра полностью все задания, он получает максимум 50% баллов от исходного, т.е. необходимый минимум для допуска к экзамену. При этом все должно быть выполнено без претензий к качеству. На экзамене он фактически получает на 1 традиционный балл ниже, т.е. вместо 5 – 4 и т.п., и. ниже 3 сдавать не имеет права (справедливость торжествует, порок наказан).

Ведение архива отчетных документов

Необходимо сохранять текущие документы по единицам контроля - отчеты и исходные данные (например, исходные кода программного проекта). Для целей временного хранения и систематизации имен документов в каждой БД поддерживается **хранилище.** Оно используется не как архив, а как средство перекачки и систематизации документов в личный архив ведущего преподавателя дисципльны.

Степень централизации и затраты на обслуживание

В целом информационная система университета является централизованной. Значительная часть функционала реализована в виде web-приложений, сопровождение которых ведется службой учебного заведения (ЦИУ). Для аналогичной реализации проекта имеются важные сдерживающий факторы внедрения:

- информационная безопасность. Обычно доступ к корпоративным ресурсам осуществляется с рабочего места преподавателя, здесь же речь идет об учебной аудитории или лаборатории, в которой есть вероятность оставить без контроля «залогиненную» программную компоненту;
- объем архивных данных и текущий трафик;
- отсутствие постоянного доступа к компьютеру. Несмотря на тотальную компьютеризацию, возможны ситуации, когда компьютера просто нет под рукой, например, при проверке посещаемости в начале лекции, либо при приеме заданий в учебной аудитории и т.п.;

- разделение ответственности и обязательный регламент. До сих пор ответственность за текущий документооборот по учебному процессу несли преподаватели, разделение этой ответственности с централизованными службами в массовом масштабе может породить ненужные трения и психологический дискомфорт;
- свобода выбора преподавателя - нормативные документы определяют рамочные конструкции для вычисления рейтинга по основным схемам учебных дисциплин, а преподаватель в рабочей программе дисциплины наполняет их конкретным содержанием. С этой позиции введение единообразия не имеет под собой необходимых оснований.

Автоматизация «сверху» не смотрится по общесистемным соображениям. Она может породить неэффективного, слабо управляемого монстра. Предлагаемый вариант:

- независимые БД для групп преподавателей, администрирование БД одним из преподавателей (куратором)
- централизованный хостинг БД на сервере БД учебного заведения или серверах подразделений

Примечание. Часть приведенного материала может быть отнесена к бизнес-процессам предметной области. Но поскольку проект меняет существующие бизнес-процессы, то материал имеет отношение к видению проекта.

Границы проекта

В проекте не предусмотрены:

- средства миграции данных в другие подсистемы управления учебного процесса. Используются только логины ЦИУ НГТУ для доступа студентов (возможно, и преподавателей).
- мета-уровень описания структуры учебного процесса. Возможные схемы расчета рейтинга “зашиты” в программный код

Перечень пользователей проекта

Каждый тип пользователей имеет свое оригинальное приложение для работы:

- **Преподаватель -** ввод и редактирование данных об успеваемости и посещаемости, вывод отчетов, сохранение и выгрузка документов (отчетов и исходников) в хранилище. Имеется desktop и мобильное (Android) приложения.
- **Куратор -** преподаватель, выполняющий конфигурирование отдельной БД под структуру учебного процесса - ввод данных о группах, студентах, предметах, единицах контроля, рейтингах, преподавателях и разрешениях. Архивирует и восстанавливает БД. Использует отдельное desktop-приложение, автономное, либо встроенное в приложение преподавателя.
- **Студент -** просмотр данных о собственном рейтинге,сохранение документов (отчетов и исходников) в хранилище
- **Администратор сервера БД -** работа со списком БД на сервере (создание и удаление), вход в режиме преподавателя и куратора, просмотр и редактирование лога (списка фатальных ошибок)

Бизнес-аналитика предметной области

Комментарий к документу. Бизнес-аналитика предметной области используется для описания бизнес-процессов, связанных с реальными сущностями (предметами, персоналиями, документами и т.п.). Этим она отличается от **системной аналитики,** которая описывает внутреннее представление, процессы и интерфейсы в **программной системе (ПС).** Например, если клиент покупает билет в кассе, то его взаимоотношения с кассиром являются предметом бизнес-аналитики, но не системной аналитики.
Бизнес-аналитика необходима для выявления **сущностей** и их отношений, а также для описания процессов в предметной области.

Балльно-рейтинговая система предполагает итоговую оценку успеваемости по 100-балльной шкале. В соответствии с учебным планом **группа студентов** изучает **предмет,** включающий в себя **единицы контроля,** т.е. учебные мероприятия, по которым дается **оценка в баллах**. Каждая единица контроля имеет **нормативную оценку,** сумма нормативных оценок по предмету должна быть равна 100…. и.т.д.

Глоссарий

Комментарий к документу. Глоссарий выносятся все термины, обозначающие сущности (бизнес-сущности, архитектурные, конструктивные) для единообразия всех документов и коммуникаций. Сюда же могут относиться термины, касающиеся архитектуры, поведения, алгоритмов и других характеристик ПС.

Недопустимо, когда в документах используются различные термины для обозначения одной и то же сущности (например, заказ,заявка) или выражения типа “список заказов в колонке слева” вместо “список необработанных заказов”.

Допускается использование не принятых в данной области терминов, жаргонных терминов или метафор для обозначения сущностей. Например, “мешок подарков” вместо “комплект заказов для перевозки”.

Расшифровка сущности требуется, если она не является общепринятой. Например, “пассажир - лицо, использующее такси для поездки”. Наоборот, в расшифровке есть необходимость для определения свойств сущности, *специфической для данного проекта*.

Предмет – оригинальная дисциплина учебного плана, читаемая в **одном** семестре и имеющая **одну** итоговую оценку

Единица контроля – составная часть предмета, по которой выносится оценка, включающаяся в рейтинг

Рейтинг - набор данных для *одной учебной группы и одной дисциплины*

Хранилище - область данных сервера для временного хранения ограниченного количества файлов отчетов и «исходников» до момента их скачивания преподавателем в собственный архив

ЦИУ - Центр информатизации университета. Сервис и данные авторизации используются для авторизации студентов (обязательно) и преподавателей (по выбору)

Подгруппа - выполнение некоторых видов единиц контроля происходит в 2-х подгруппах, для которых устанавливаются разные сроки сдачи

Показатель качества исполнения единицы контроля - набор качественных характеристик (включение по принципу да-нет), за которые снимается (начисляется) фиксированный процент к нормативному баллу. Зашиты в ПС (не содержатся в БД)

Срок сдачи - устанавливается для единиц контроля, для которых предусмотрено формальное вычисление балла

Отчет - документ, загружаемый в хранилище, отчет о выполнении единицы контроля

Исходник - документ или архив, загружаемый в хранилище, содержит

Куратор - преподаватель, имеющий права на редактирование и архивирование мета-данных в БД рейтингов (структура предмета, списки групп и преподавателей, разрешения)

Администратор сервера БД - лицо, имеющее право создавать и уничтожать БД на сервере БД, а также просматривать лог ошибок

БД рейтингов - отдельная БД для нескольких предметов, студенческих групп, преподавателей и рейтингов. Создается для цикла родственных дисциплин

Сервер БД рейтингов - сервер хранит несколько независимых БД рейтингов, а также БД с описанием этих БД

Преподаватель - без комментариев

Группа - без комментариев

Бригада - без комментариев

Вариант задания - без комментариев

Модель классов предметной области

Комментарий к документу. Модель классов предметной области описывает представление в информационной системе сущностей предметной области и их постоянных отношений. Тестирование модели состоит в проверке возможности хранения и извлечения данных для основных прецедентов. В дальнейшем модель преобразуется в другие модели ТП анализа и проектирования:

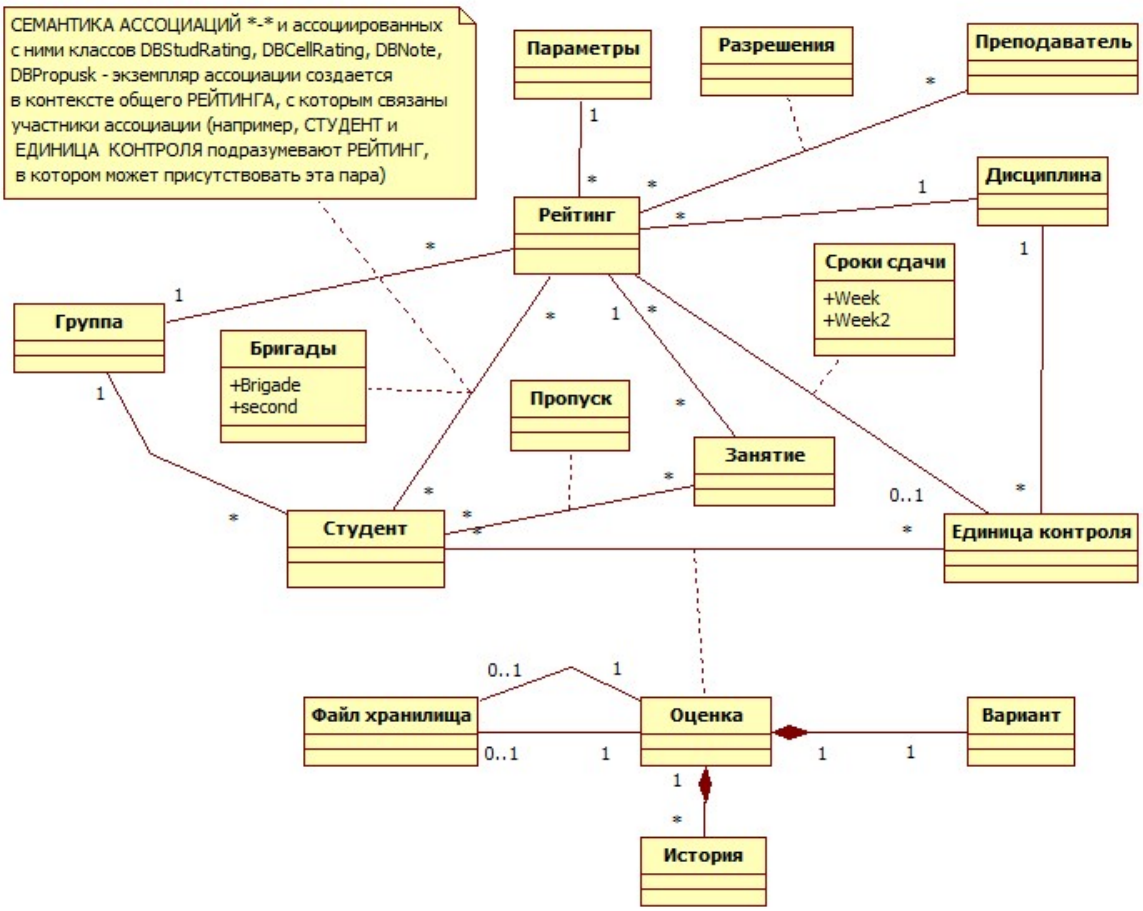
- модель БД серверного приложения
- модели классов бизнес-объектов

Важно, что модель классов предметной области описывает содержание (статику) ПС, а не ее поведение (динамику). Поэтому сущности типа **“оформление заказа”, “приобретение билета**” сюда не относятся.

В модель предметной области не включаются сущности и отношения:

- временная сущность, не сохраняемая в ПС. Пример: **отчет,** если он только формируется, и не хранится в архиве отчетов
- отношения между сущностями, если они не сохраняются в ПС, не используются в описании поведения ПС (прецеденты). Пример: зритель, покупающий билет в кассе, взаимодействует с ПС опосредованно через кассира, данные о нем не сохраняются, поэтому отношения “клиент - кассир (касса, смена)” в модели отсутствуют.

При наличии различных вариантов реализации отношения оно должно устанавливаться между сущностью и **интерфейсом** или **абстрактным классом.** Пример: билет может быть продан через кассу, забронирован через интернет, либо оплачен через платежные системы.Отношение устанавливается между сущностями **билет - способ продажи** (интерфейс или абстракция), от которой наследуются сущности **“через кассу”, “бронирование”, “через платежную систему”**

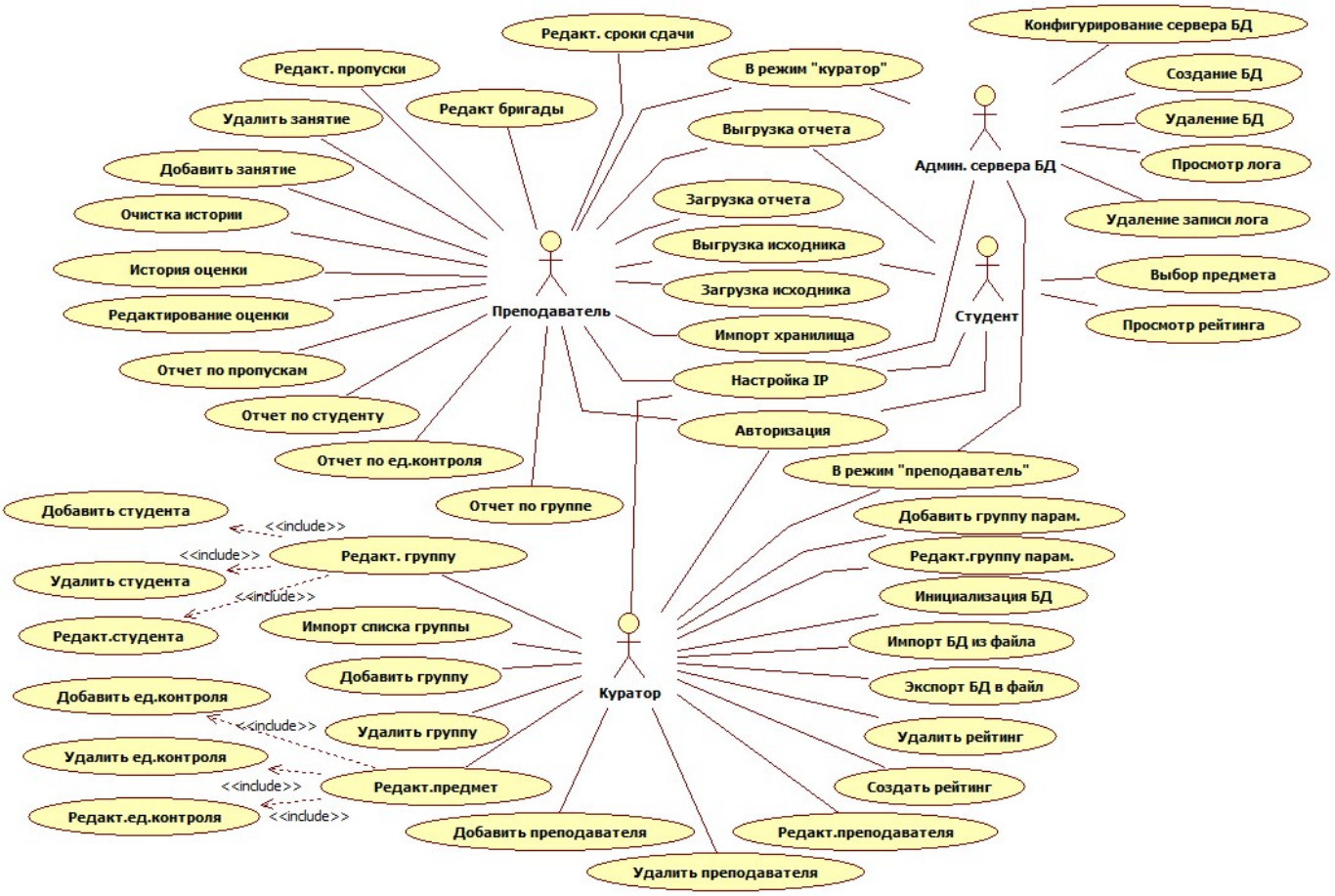


Описание модели

1. **Группа-студент, Дисциплина-единица контроля** - обычная двухуровневая иерархия.
2. **Рейтинг -** основная сущность модели, соответствующая паре **группа - дисциплина,** при создании рейтинга создаются необходимые отношения и ассоциированные с ними данные (классы) **студент ,рейтинг - бригада, единица контроля,рейтинг - срок сдачи, студент,единица контроля,рейтинг - оценка, студент,занятие,рейтинг - пропуск**
3. С отношением **преподаватель-рейтинг** связан ассоциированный класс **разрешения**
4. С рейтингом связан набор **параметров,** используемых для его вычисления

Модель прецедентов. Сценарии

Комментарий к документу. Прецедент - ограниченное по времени и функционалу (атомарное) взаимодействие пользователя с системой. Следует обращать внимание на полноту списка прецедентов, дабы не допускать пропуска видов работ, выполняемых пользователями.



Разработка требований

Комментарий к документу. Классификация требований.
Бизнес-требования – цели создания системы (организация, заказчик), получаемый эффект, образ и границы проекта
Требования пользователей – описания возможных действий пользователей и их поведения (варианты использования (прецеденты), сценарии)
Функциональные требования – перечень реализуемого функционала с описанием основных параметров и характеристик (…должен…, может быть элементом нескольких прецедентов, не отраженным текущем уровне детализации сценария)
Системные требования – перечень требований, распространяющихся на всю систему в целом (ПС = ПО + оборудование + пользователи)
Бизнес-правила – законы, постановления, юридические нормы, сложившаяся практика
Атрибуты качества – эффективность, надежность, простота использования и т.п. Основное требование - должны быть сформулированы в виде **количественных оценок,** допускающих проверку и тестирование. Виды атрибутов качества:

- доступность (% или время доступа с учетом сбоев, отказов, установки)
- эффективность (затраты на обеспечение производительности)
- гибкость (расширяемость, масштабируемость)
- целостность (безопасность)
- надежность (гарантии работы без сбоев)
- устойчивость (к сбоям, способность к восстановлению)
- удобство использования (useability)
- удобство эксплуатации
- мобильность, портируемость
- повторное использование (кода, данных)

Ограничения, следующие из бизнес-правил, условия, при которых невозможно выполнение прецедентов

Форма документа описания требований должна соответствовать его дальнейшему использованию в ЖЦ ПО. Возможные варианты:

- **Документ** в “тяжеловесных” технологиях проектирования ПО. Выполняется в полном объеме (например, в соответствии с “Спецификация требований к ПО” на основе IEEE Standard 830-1998)
- **Техническое задание** на внешнее исполнение (аутсорсинг). Включает в себя структуру БД, системные требования, бизнес-правила, требования к интерфейсам (в т.ч. графическим), развернутые сценарии.
- **Иерархический справочник** в гибкой технологии разработки ПО. При наличии других документов нет необходимости расписывать сценарии (прецеденты). Необходима общая иерархическая БД фактов, касающихся ПС. Признаки классификации могут быть разными: особенности системы, варианты использования (use cases), режим работы, классы пользователей, стимулы, реакции, классы объектов или функциональной иерархии. Желательно использовать именованные теги вида **правила.рейтинг.сумма.** Один и тот же факт может присутствовать в разных ветках классификации (необходимы взаимные ссылки)

Бизнес-правила

правила.рейтинг.параметры.начало семестра - текущая неделя отсчитывается от этого параметра

правила.рейтинг.параметры.штраф_за_пропуск - балл, снимаемый за пропуск занятия (kp)

правила.рейтинг.параметры.%за_показатель - начисляемый или снимаемый % за *показатель качества* исполнения единицы контроля (dq)

правила.рейтинг.параметры.%за_неделю - процент, снимаемый за одну неделю просрочки. При досрочной сдаче - симметрично добавляется (dw)

правила.рейтинг.параметры.недель_просрочки - интервал в неделях, в течение которого снимается процент, в дальнейшем - остается на достигнутом уровне

правила.рейтинг.сумма - сумма баллов рейтинга по обязательным единицам контроля должна бы 100. Для учета индивидуальных внеплановых заданий и бонусов вводятся единицы контроля с весом 0

правила.рейтинг.пропуск - за пропуск занятия из общей суммы снимается балл рейтинга, указанный в *параметры.рейтинг.параметры.штраф_за_пропуск*

правила.рейтинг.показатель_качества_исполнения - устанавливается как логическое значение (наличие/отсутствие), предусматривает как увеличение, так и уменьшение балла на фиксированный процент, например, “+сложность”,”-оформление” (pi)

правила.рейтинг.единица_контроля.норматив -каждой единице контроля назначается нормативное количество баллов

правила.рейтинг.единица_контроля.субъективная - балл в единице контроля ставится преподавателем по собственным оценкам (неформально) в пределах установленного значения (экзамен - 40 баллов), либо по критериям, не включенным в систему

правила.рейтинг.единица_контроля.формальная - балл в единице контроля выставляется системой по формуле расчета

правила.рейтинг.единица_контроля.формула - N0 - (w - w0)*dw + dq*∑pi

Атрибуты качества

качество.код.повторное использование - весь код, за исключением слоев представлений (экранов, View) и является независимым от приложения и может быть повторно использован. Контроллер приложения “преподаватель” полностью отделен от представления и используется в desktop и android-приложениях

качество.доступность.автономно - при отсутствии соединения с сервером приложение преподавателя может работать с локальными копиями рейтингов, которые открывались при работе с сетью. При повторном входе и наличии соединения с сервером внесенные изменения автоматически синхронизируются с сервером БД

качество.доступность.масштабируемость.сервер_БД - приложение имеет настройки на сервер БД. Сервер БД содержит список независимых однотипных БД

Функциональные требования

функция.оценка.история - должна хранить все записи об изменении оценки и обеспечивать просмотр

функция.оценка.история.очистка - очистка истории по всему рейтингу с оставлением последней оценки

функция.рейтинг.имя - имя рейтинга = название предмета <пробел> название группы

функция.хранилище.объем - для каждого рейтинга должно выводиться кол-во файлов отчетов и исходников (в сумме) и их объем.
функция.хранилище.скачивание - преподаватель скачивает все файлы отчетом и исходников для выбранного рейтинга “одним кликом” в выбранный каталог, при скачивании файлы из хранилища удаляются
функция.хранилище.загрузка.разрешение - загрузку файлов в хранилище могут выполнять преподаватель и студент
функция.хранилище.скачивание.путь - в выбранном каталоге для скачивания создается каталог с именем рейтинга
функция.куратор.БД.экспорт - куратор может сохранить содержимое БД рейтингов в файл
функция.куратор.БД.импорт - куратор может загрузить содержимое БД рейтингов из
функция.куратор.БД.инициализация - куратор инициализирует БД рейтингов, старая БД уничтожается, создается новая структура БД
функция.преподаватель.разрешения - преподаватель имеет список разрешений на просмотр/редактирование рейтингов, устанавливаемый куратором.
функция.куратор.разрешения - куратор в режиме “преподаватель” имеет доступ ко всему списку рейтингов
функция.куратор.единица_контроля.порядок - порядок следования единиц контроля при выводе редактируется куратором
функция.куратор.список_группы - список группы студентов может быть загружен из текстового файла, полученного копированием соответствующей формы ЦИУ через clipBoard.Цифровые номера зачетных книжек при чтении файла пропускаются
функция.вывод.студент - при выводе списков и текстовых полей при просмотре и редактировании рейтинга отчество студента не выводится.
функция.единица_контроля.тип - набор типов единиц контроля зашит в приложения (не хранится в БД)
…….. и т.д.

Отчеты

отчеты.формат - необходимо представление всех отчетов в форматах html, pdf и интерактивная форма
отчеты.интерактивная_форма - вывод отчета в виде экранной формы с возможностью прокрутки и позиционирования к основной форме через клик по ячейке таблицы с установкой параметров выбранной ячейки (например, студент - единица контроля)
отчеты.виды - по группе (оценки), по единице контроля, по студенту, пропуски (по группе)

Графический интерфейс

GUI.мобильный.отчет - должен быть реализован в интерактивной форме. При ограничении размеров экрана прокрутка должна сопровождаться **выделением цветом** выбранных строки и столбца
GUI.desktop.отчет.интерактивная_форма - прокрутка таблицы отчета должна производиться с сохранением в панели просмотра заголовков строк и столбцов
GUI.desktop.подсказки - клик по правой кнопкой мыши по любой кнопке формы сопровождается выводом подсказки о ее функции
GUI.форма.title - каждая форма в заголовке содержит основные данные о позиционировании - ip сервера, имя БД, фамилию студента, название предмета

Архитектурное проектирование

Комментарий к документу. Архитектурное проектирование в наибольшей степени представляет “штучный продукт”, основывается на личном опыте и кругозоре разработчиков. Архитектура включает в себя:

- **значимые решения** по поводу организации ПС
- **структурные элементы и их интерфейсы**, при помощи которых компонуется система
- **поведение** - взаимодействие между этими элементами
- **компоновка элементов** в иерархию подсистем
- **стиль архитектуры** который направляет эту организацию

Архитектурное описание является **многомерным,** т.к. описывает ПС с различных точек зрения и **минимально избыточной** (добавление данных не вносит качественных изменений в описание ПС, удаление ведет к потере существенных качеств)

Архитектурными сущностями являются элементы процесса проектирования (по SWEBOK):

- ключевые идеи и концепции: абстрагирование, связность и соединение, модульность и декомпозиция, инкапсуляция, разделение интерфейса и реализации
- параллелизм
- контроль и обработка событий
- распределение компонентов
- обработка ошибок и исключительных ситуаций и обеспечение отказоустойчивости
- взаимодействие и представление (MVC)
- сохраняемость данных (доступность «долгоживущих» данных)
- виды представления архитектуры - структурное, поведенческое, логическое, физическое, реализация кода
- архитектурные стили
- шаблоны проектирования
- методы проектирования: нисходящее проектирование, модульное, абстрагирование, итеративность,функционально-ориентированное или структурное проектирование, объектно-ориентированное проектирование, проектирование на основе структур данных, компонентное проектирование

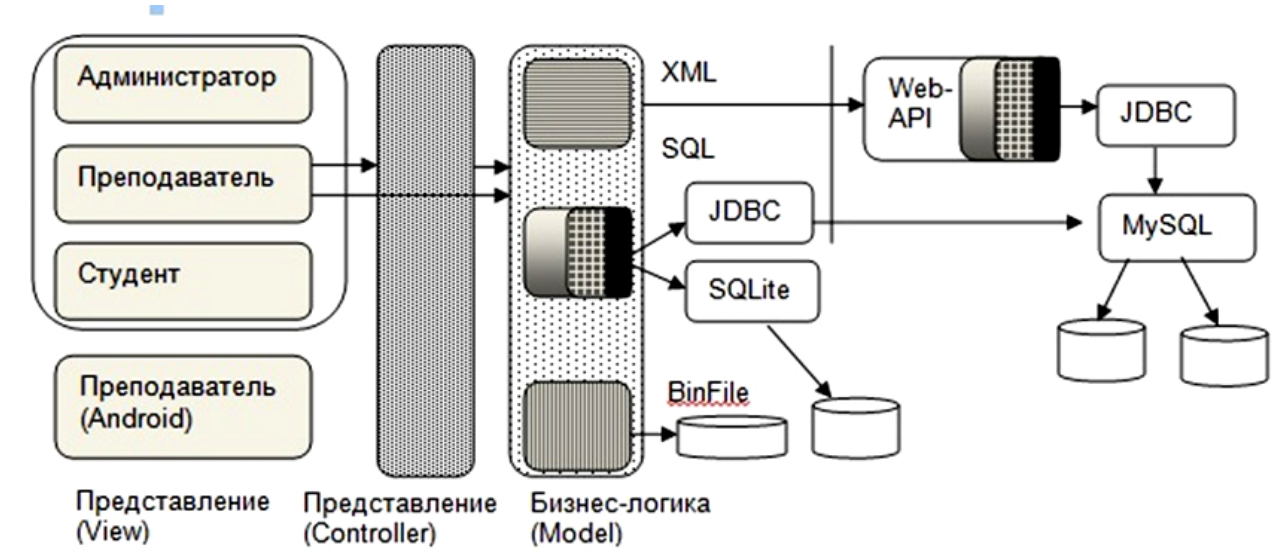
Ключевые идеи

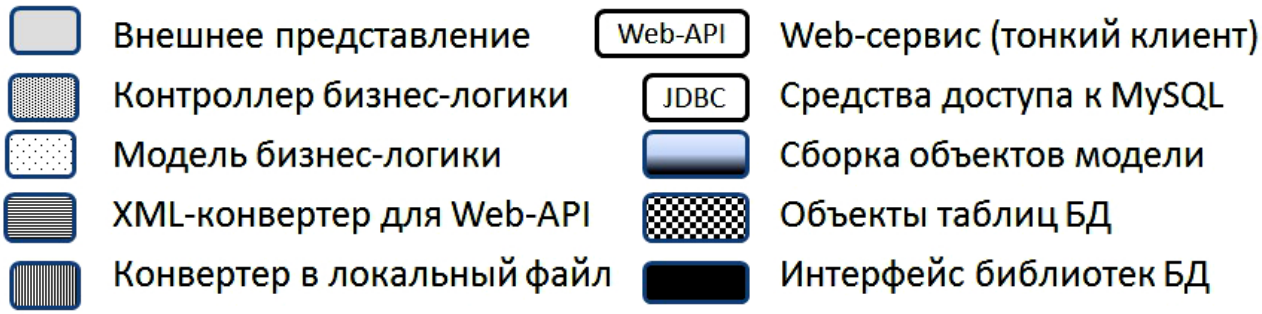
Разделение кода на функциональные слои:

- представление (View)
- поведение (Controller)
- бизнес-модель, бизекс-объекты (Model)
- табличные объекты БД (DAO)
- коннектор и библиотека доступа к СУБД
- сервер БД

Возможность реализации толстого и тонкого клиента на основе базового класса бизнес-модели, имеющего 3 реализации интерфейса получения бизнес-объектов:

- конструирование бизнес-объектов из табличных объектов DAO (толстый клиент)
- XML-сериализатор для Web API - (тонкий клиент)
- сериализатор в двоичные файлы (для локальных копий рейтинга)





Основными архитектурными компонентами являются:

- уровень представлений – реализация внешних форм клиентских приложений;
- контроллер бизнес логики – алгоритмы поведения для всех клиентов приложения “преподаватель”;
- модель бизнес-логики (классы бизнес-объектов) – внутреннее представление рейтингов, групп, студентов, оценок и т.п. и их основных алгоритмов;
- средства сборки бизнес-объектов из табличных объектов БД;
- конвертеры бизнес-объектов в двоичные файлы и в XML-формат;
- табличные объекты БД (DAO – Data Access Objects);
- интерфейс к библиотекам JDBC и SQLite;
- серверные компоненты web-сервиса (сервлеты).

Параллелизм. Синхронизация

Специальных требований к параллелизму не предъявляется. В представлениях (View) исполнение продолжительных операций должно проходить в фоновом режиме, при этом элементы управления не должны быть блокированы, но и не позволяют выполнять действия над данными, обрабатываемыми в фоновом режиме (“мягкая” блокировка GIU)

Взаимодействие и представление (MVC)

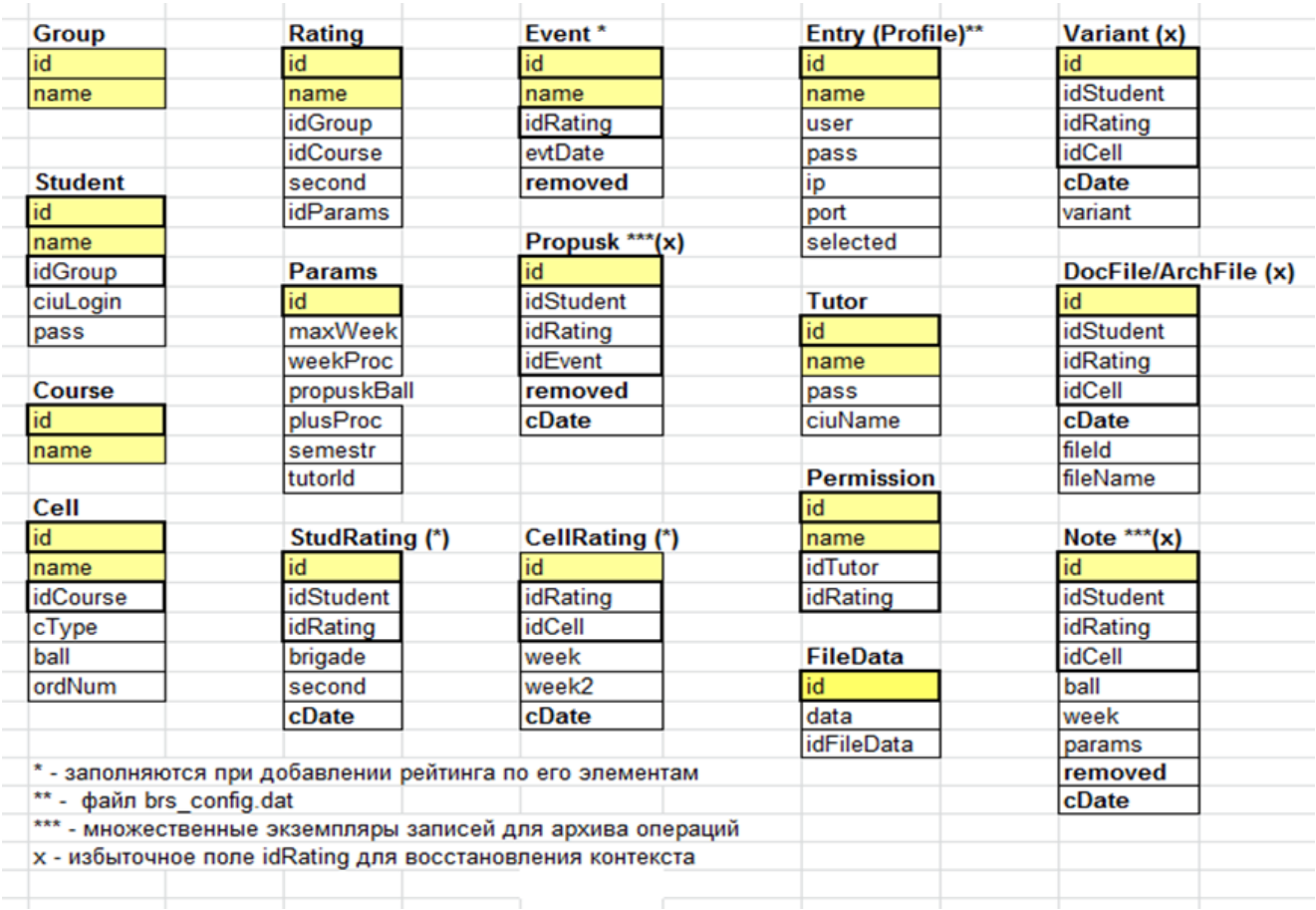
Все приложения используют уровни бизнес-объектов и ниже, т.е. общую модель представления данных. Все приложения, кроме “преподаватель” совмещают функции View+Controller в соответстствующих классах, т.к. их поведение является уникальным. Приложение “преподаватель” имеет **полностью разделеные** компоненты View и Controller по причинам:

- имеется две версии приложения с идентичным поведением - desktop и мобильное (Android)
- при отсутствии соединения с БД приложения должны использовать источники данных - локальные копии рейтингов. Этот функционал прозрачен для уровня представлений и реализован в контроллере. Последний управляет переключением и синхронизацией источников данных

Проектирование (design)

Комментарий к документу. Проработка решений, принятых при проектировании архитектуры. Разрабатывается структура кода, относящаяся к внутренним подсистемам, сервисам, форматы, протоколы и т.п.

Структура БД



Group
id
name

Student
id
name
idGroup
ciuLogin
pass

Course
id
name

Cell
id
name
idCourse
cType
ball
ordNum

Rating
id
name
idGroup
idCourse
second
idParams

Params
id
maxWeek
weekProc
propuskBall
plusProc
semestr
tutorId

StudRating (*)
id
idStudent
idRating
brigade
second
cDate

Event *
id
name
idRating
evtDate
removed

Propusk ***(x)
id
idStudent
idRating
idEvent
removed
cDate

CellRating (*)
id
idRating
idCell
week
week2
cDate

Entry (Profile)**
id
name
user
pass
ip
port
selected

Tutor
id
name
pass
ciuName

Permission
id
name
idTutor
idRating

FileData
id
data
idFileData

Variant (x)
id
idStudent
idRating
idCell
cDate
variant

DocFile/ArchFile (x)
id
idStudent
idRating
idCell
cDate
fileId
fileName

Note ***(x)
id
idStudent
idRating
idCell
ball
week
params
removed
cDate

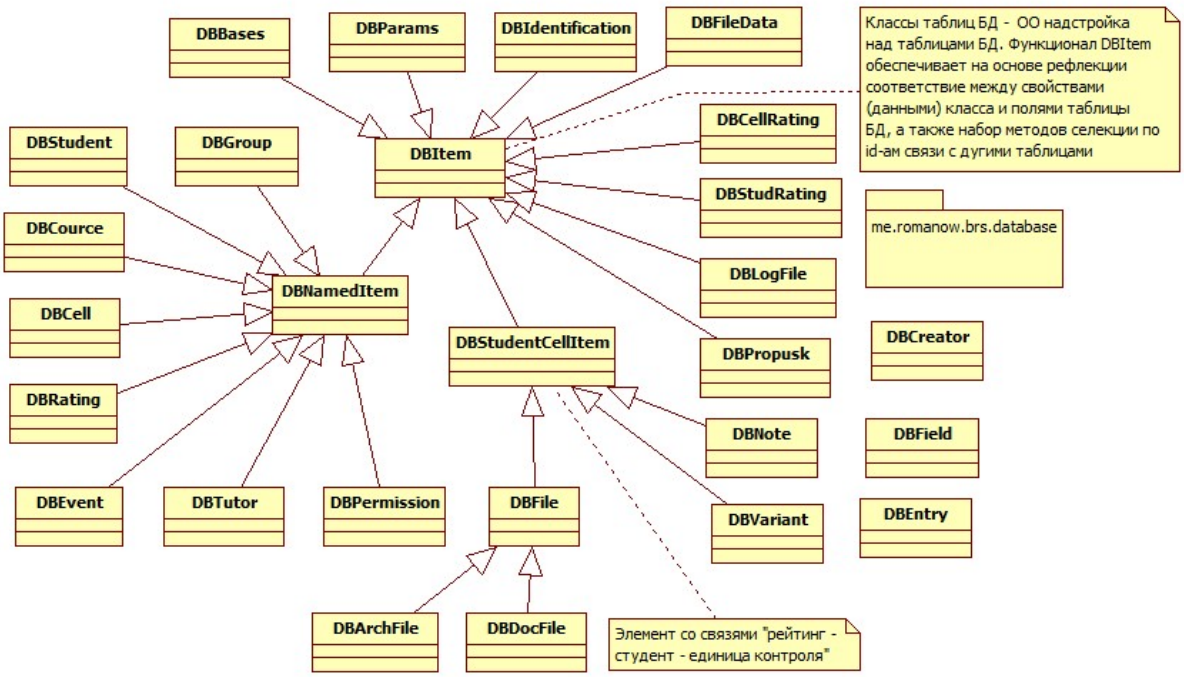
* - заполняются при добавлении рейтинга по его элементам
** - файл brs_config.dat
*** - множественные экземпляры записей для архива операций
x - избыточное поле idRating для восстановления контекста

Структура программного кода

Ядро (core) - уровни контроллера (“преподаватель”), бизнес-объектов, табличных объектов, коммуникации, коннекторы к БД, интерфейсы - повторное использование кода

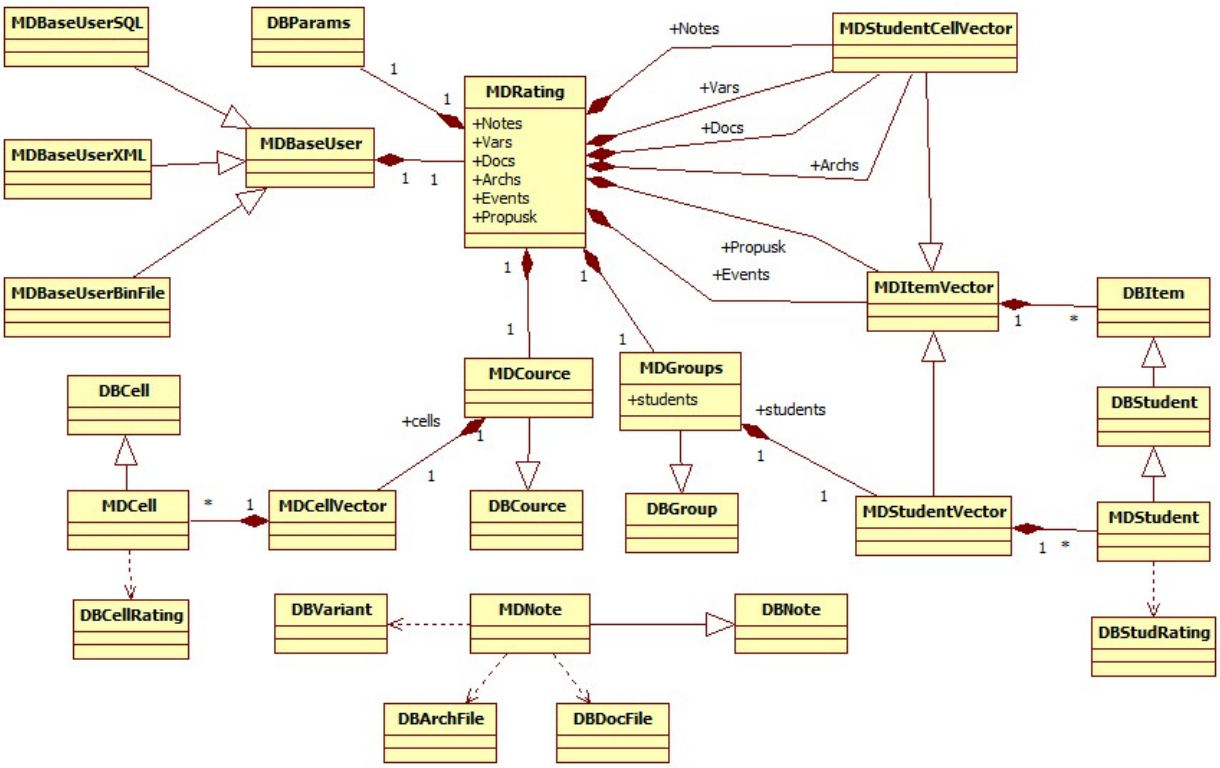
Компонента	Пакет	Классов	Строк	Назначение
Core (ядро)	brs	2	325	Параметры настройки приложений
	brs.connect	4	828	Коннекторы к БД
	brs.model	29	3412	Бизнес-объекты
	brs.database	34	1207	Табличные объекты БД
	brs.ftpclient	4	880	ftp-клиент хранилища - не используется
	brs.controller	3	646	Контроллер бизнес-логики "преподаватель"
	bts.xml	8	244	Команды-ответы протокола WebAPI
	brs.interfaces	5	143	Интерфейсы компонент
	brs.cui	5	131	Объекты протокола ЦИУ
Android	brs.view	26	3755	Уровень представления (экраны)
	brs.sqlite	2	289	Коннектор БД SQLite (Android) - не используется
Desktop	brs.javaview	40	7748	Уровени представления и бизнес-логики (экраны) - все приложения, "преподаватель" - только представление
(все приложения)	view	1	232	Таблица с прокруткой
WebAPI	brs.web	1	684	Сервлет WebAPI для тонкого клиента

Классы табличных объектов БД (DAO - Data Access Object)



Классы бизнес-объектов

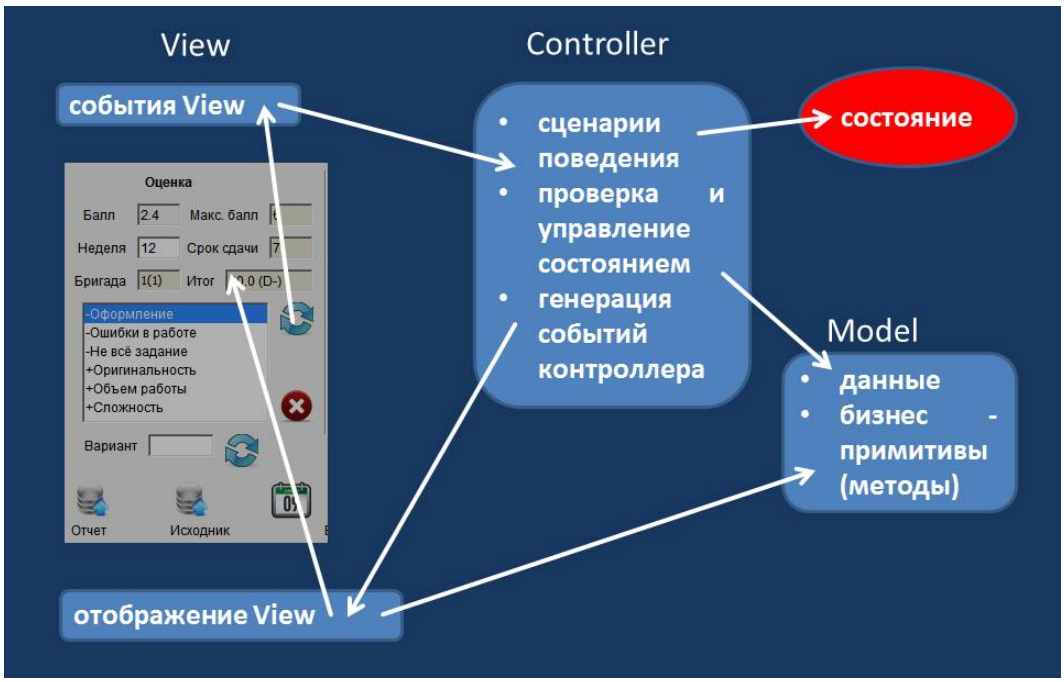
Классы бизнес-объектов составляются из данных табличных объектов, либо ссылающихся на них. Бизнес-объекты имеют наборы базовых методов и несколько уровней “развернутости” (краткий, базовый и полный). Основной бизнес-объект - рейтинг, инкапсулирующий в себя все остальные



Классы контроллер - представления (MVC)

Вариант реализации шаблона MVC. Контроллер – имитатор поведения абстрактного представления (View). Представление – тонкий клиент, лишенный модели поведения. Контроллер представление связаны двумя интерфейсам - “событий” в представлении и “команд” контроллера по вводу/выводу содержимого и отображению элементов управления в представлении. Модель - набор бизнес-объектов и бизнес-методов - в основном управляется контроллером, хотя некоторые данные могут быть получены непосредственно представлением.

Контроллер реализует варианты поведения представления, определяет возможные последовательности «нажатия кнопок» и генерирует события для изменения состояния элементов отображения. Контроллер реализует модель поведения на основе диаграммы состояний.

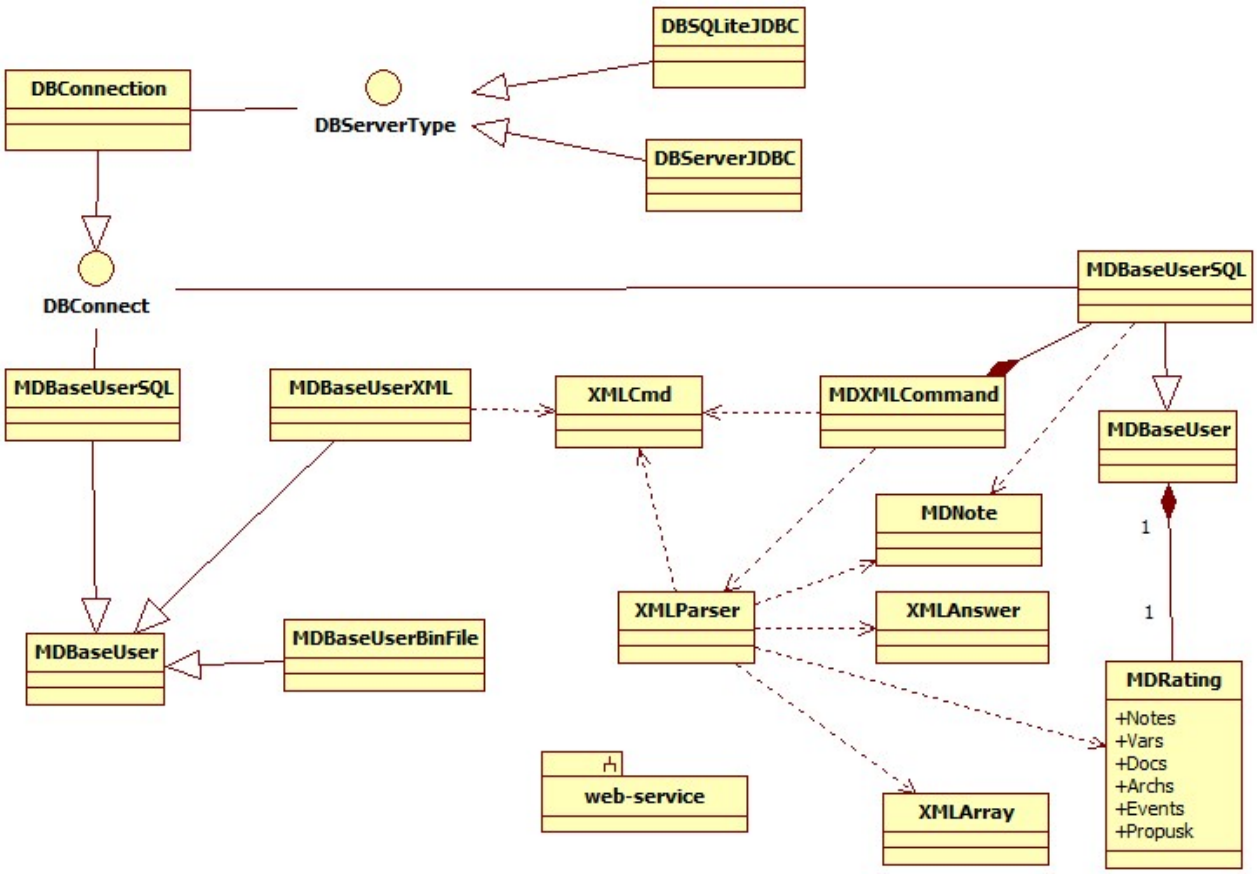


Преимущества подхода – простота реализации приложения на различных платформах (без копипаста)

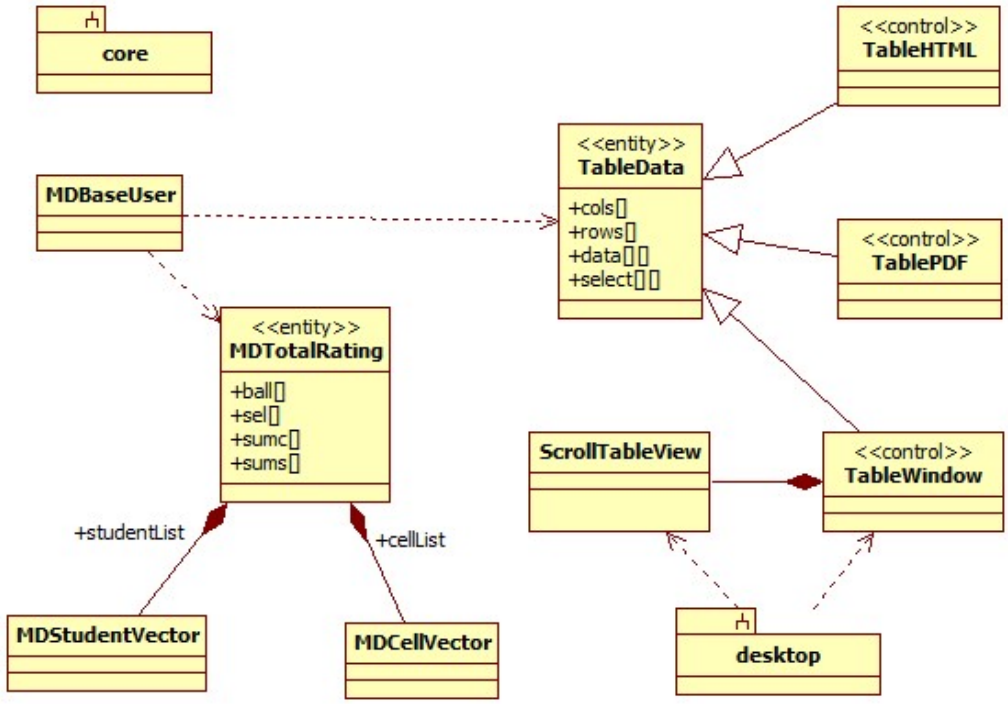
Классы коммуникаций

Классы коммуникаций обеспечивают реализацию толстого и тонкого (WebAPI) клиентов. Интерфейс DBServerType позволяет использовать два типа БД (MySQL и SQLite). Базовый класс бизнес-логики MDBaseUser имеет 3 расширения:

- MDBaseUserSQL загружает бизнес-объекты непосредственно через соединение с БД, используется в толстом клиенте
- MDBaseUserBinFile сохраняет бизнес-объект “рейтинг” в двоичном файле, используется для создания локальных копий рейтингов
- MDBaseUserXML реализует тонкого клиента с использованием WebAPI. Используется ряд классов, сериализуемых в XML, для поддержки команд и ответов протокола (XMLCmd, XMLAnswer, XMLArray). В протоколе также сериализуются бизнес-объекты (MDNote). MDBaseUserXML является клиентской частью протокола, а MDXMLCommand - серверной (Java-сервлет). Последний создает на сервере объект MDBaseUserSQL и выполняет в нем необходимые команды от имени клиента.

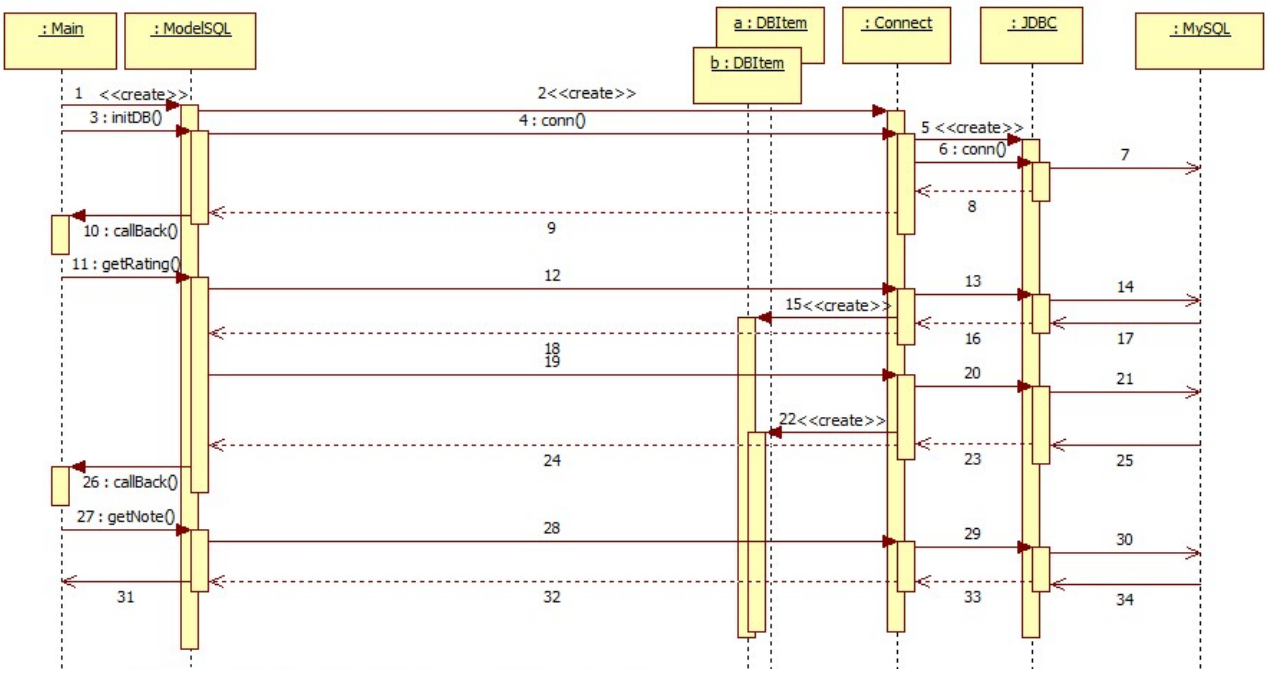


Классы формирования отчетов

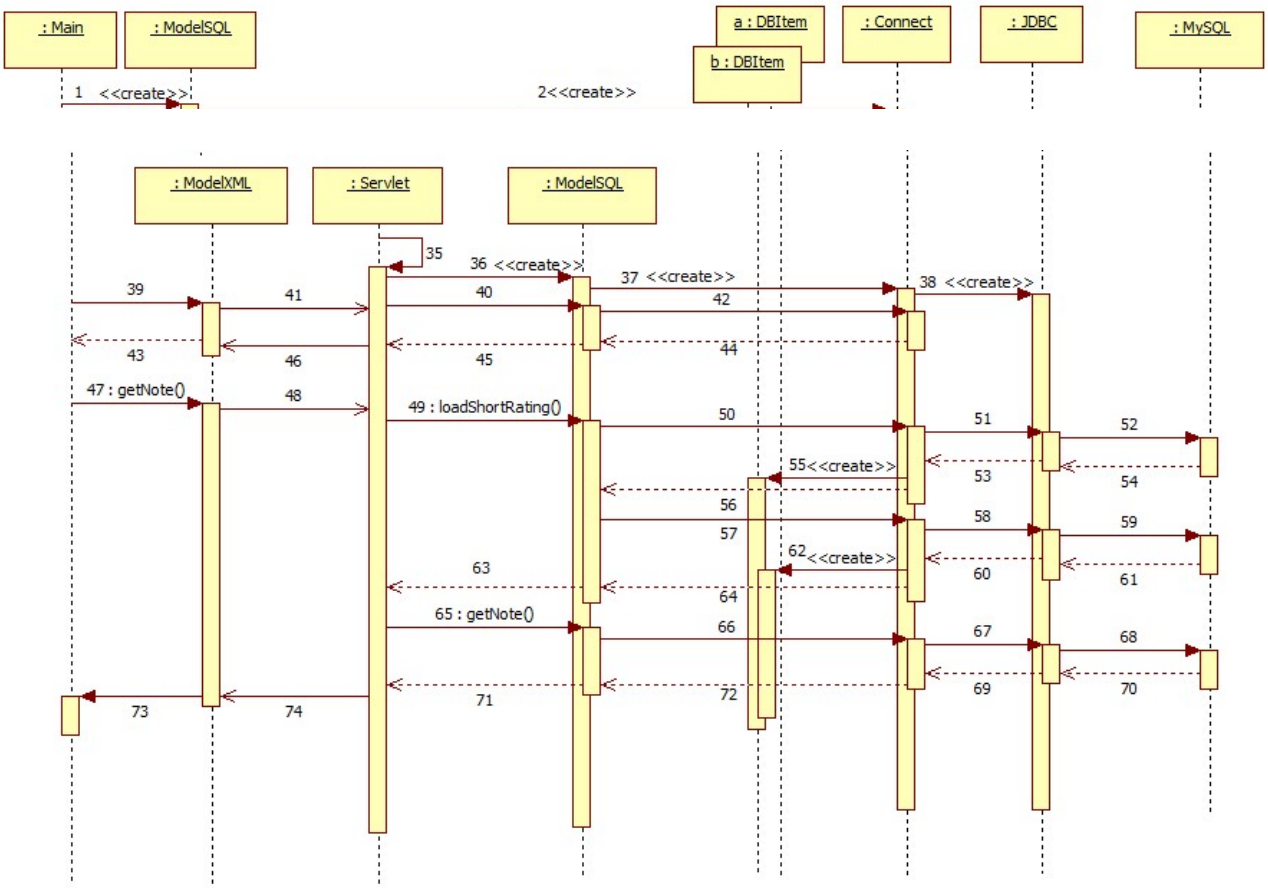


Поведение. Диаграммы последовательности для толстого и тонкого клиентов

Толстый клиент. Сетевая компонента интегрирована в библиотеку доступа к БД (JDBC)



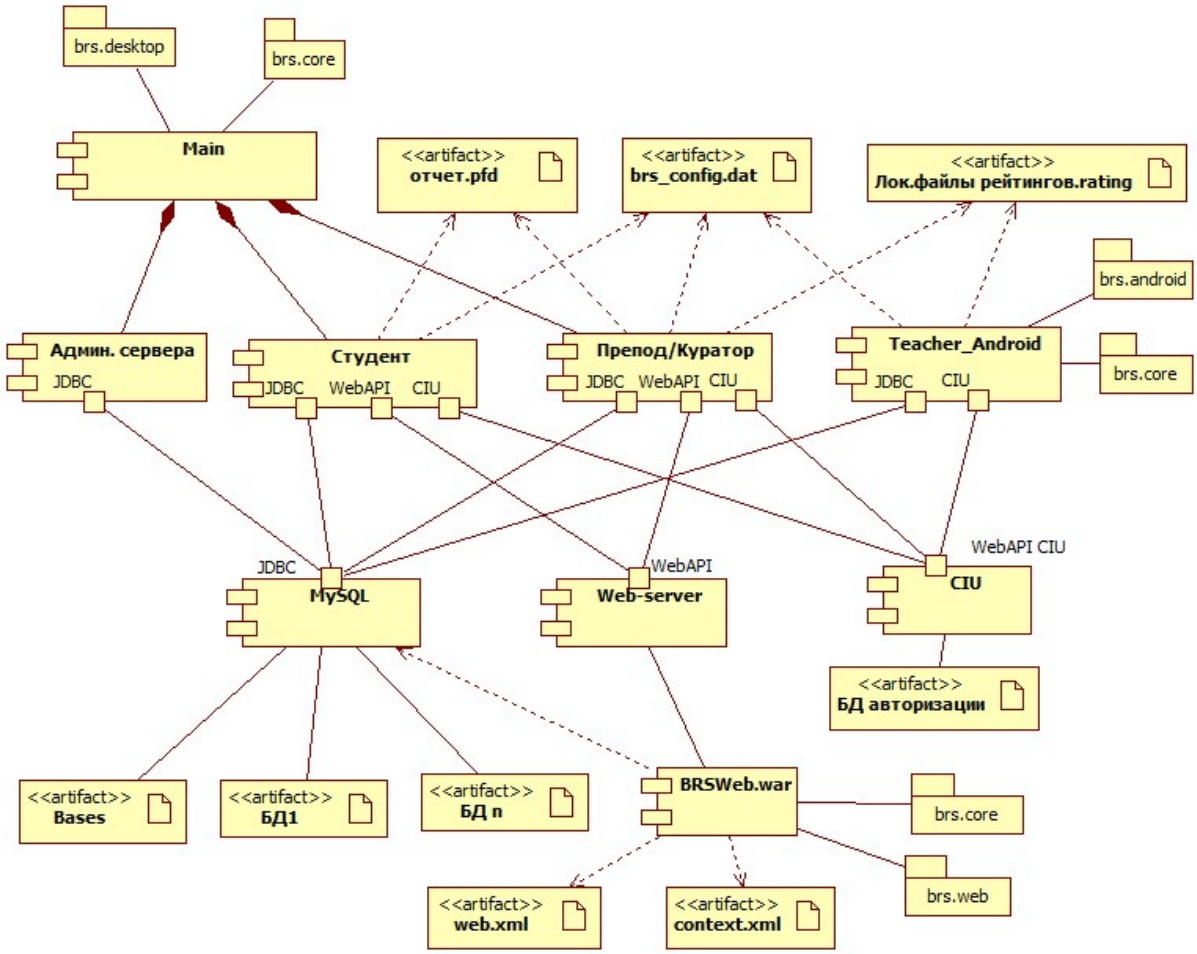
Тонкий клиент. Сетевой протокол реализован в классах ModelXML (MDBaseUserXML) и Servlet (MDXMLCommand)



Структура. Диаграммы компонентов

Диаграмма компонентов отражает структуру программных компонент проекта и их взаимосвязи, в том числе:

- структуру программного кода, вхождение пакетов исходного кода в приложения
- интерфейсы соединения приложений (клиент - сервер)
- артефакты - БД, конфигурационные файлы, отчеты, локальные копии, экспорт-импорт

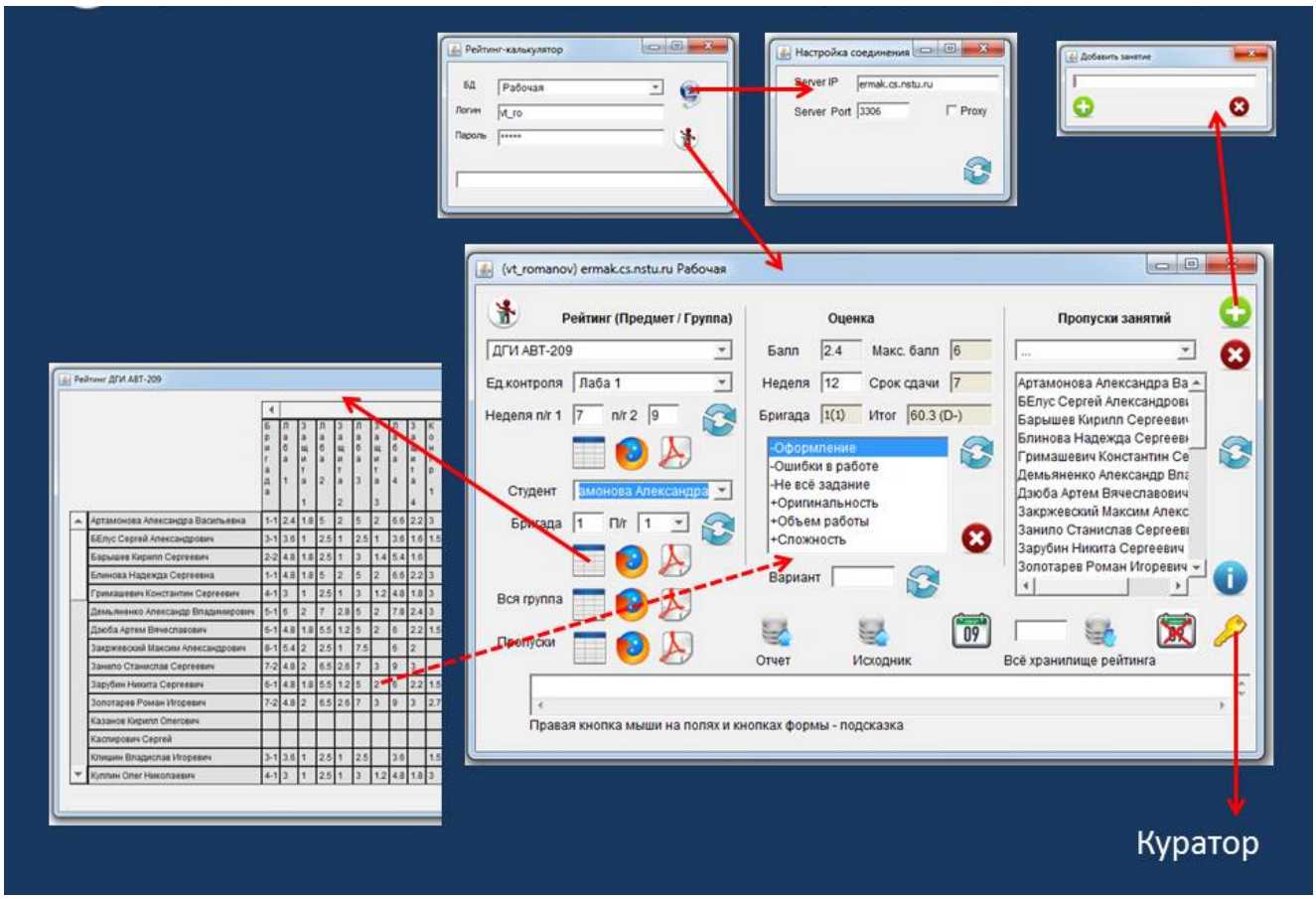


Проектирование GUI

Комментарий к документу. Структура графического интерфейса может быть представлена в виде неканонической диаграммы “экранов”. Тестирование GUI состоит в проверке реализации прецедентов по разработанным диаграммам.
При описании графического интерфейса следует отметить ключевые факторы, определяющие качество интерфейса:

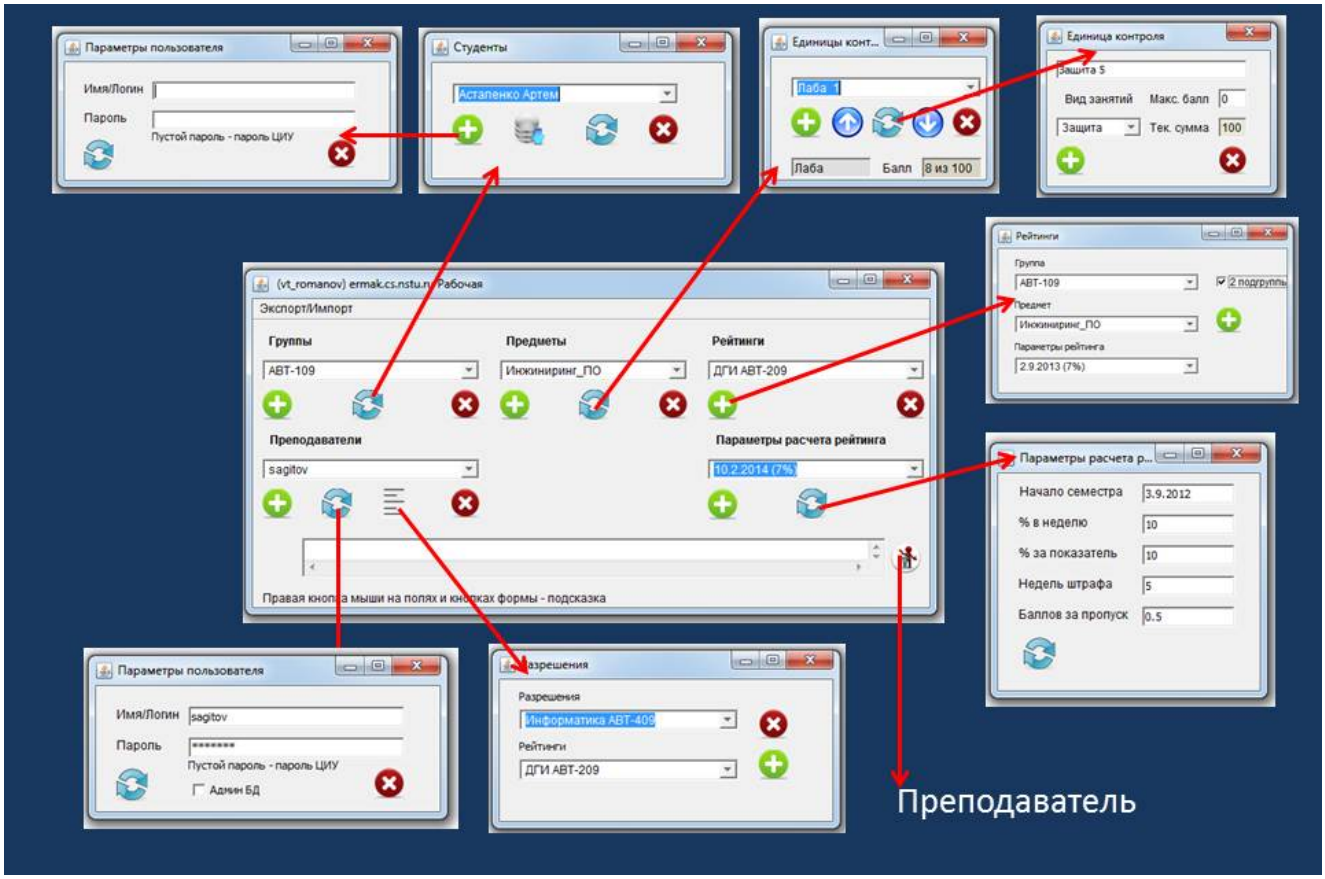
- Концепция GUI: “кабина самолета”, “главный экран с набором инструментов”, “дерево диалогов”
- Производительность при работе с интерфейсом
- Человеческие ошибки - устойчивость к ошибкам, возможность исправления, дружественность интерфейса, сообщения об ошибках
- Обучение - справочная система, ее встраивание в контекст, документация, использование метафор, “понимание” системы (следует избегать фраз типа “интуитивно понятный”)
- Субъективное восприятие - стиль, цветовая палитра, “прозрачность” интерфейса
- Запоминание, поиск, визуализация, навигация - удобство предоставления и поиска нужной информации

Desktop-приложение “Преподаватель - куратор - студент”

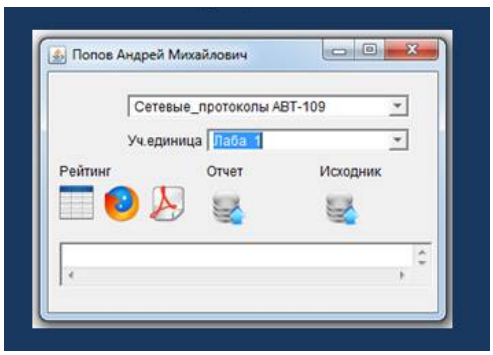


- интерфейс “кабина самолета” - полный функционал приложения на главной форме

- отчет в виде интерактивной формы с сохранением заголовков строк и столбцов при прокрутку
- клик по ячейке интерактивной формы - позиционирование к соответствующей записи (оценке) в главной форме
- клик правой кнопкой мыши по кнопкам главной формы - подсказка - назначение кнопки
- переход к форме “куратор” при наличии прав авторизации



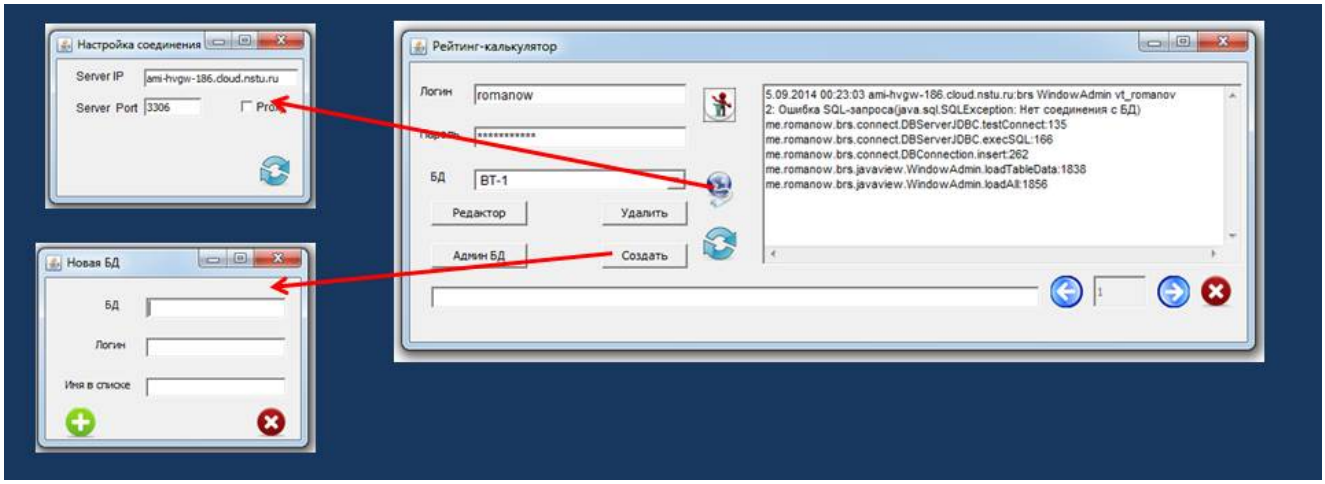
- интерфейс “главное окно” + набор диалогов под каждый вид редактируемых данных



- вход при авторизации, если введен логин/пароль студента (ЦИУ НГТУ)
- минимальный функционал просмотра данных (отчетов) + загрузки файлов

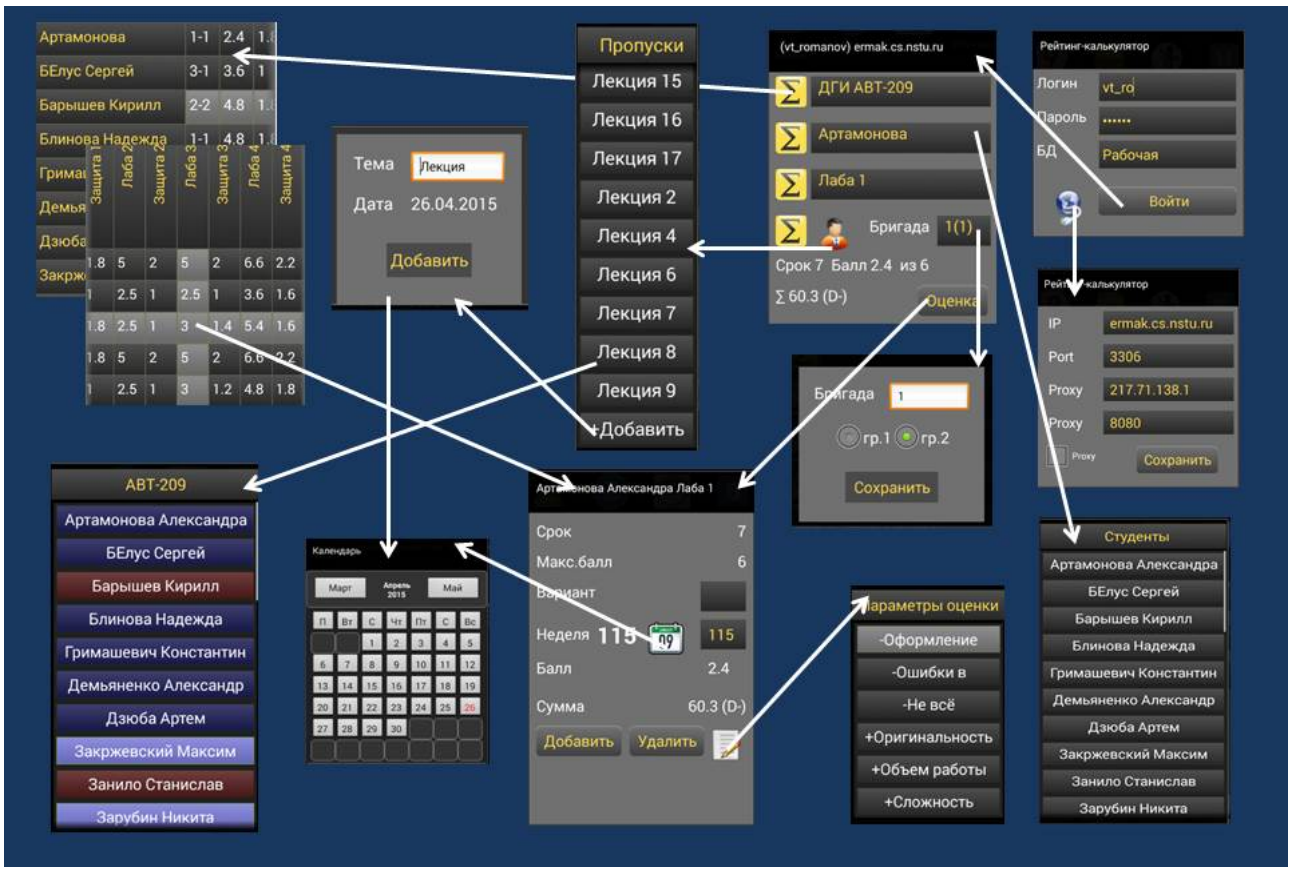
Desktop-приложение “Администратор сервера БД”

- интерфейс “главное окно” + набор диалогов



Мобильное приложение “Преподаватель”:

- ограниченный размер экрана - интерфейс “цепочка диалогов”
- контрастная сдержанная гамма “желтый-серый-черный”
- кнопки - всплывающие списки с прокруткой
- цветовое выделение “подгрупп” и “пропусков” в списке
- отчет с прокруткой по вертикали и горизонтали с цветовым выделением строки и столбца



Управление проектом. Метрика проекта. Оценка трудоемкости

Оценки по готовому проекту. Исходные данные по метрике проекта собраны в SonarCube. Базовая оценка по отраслевой статистике (COCOMO)

	core	desktop	android	web	Всего
Строк	9109	9356	4563	765	**23793**
Строк кода	7816	7980	4044	684	**20524**
Операторов	5155	4843	2287	525	**12810**
Пакетов	9	2	2	1	**14**
Классов	99	43	31	1	**174**
Методов	889	467	284	16	**1656**
Методов доступа	200	3	2	1	**206**
Файлов	94	41	28	1	**164**
Замечаний	3792	2036	1100	164	**7092**
Комментариев %	5,8	11,5	6,8	8,4	
Документированность %	7	4	2	25	
public API	880	504	298	4	
недокум. API	819	484	292	3	
Сложность:					
методов	2,7	3,4	3,6	9,1	**3,4**
классов	24,1	37,2	34.5	146	**28,5**
файлов	25,4	39	38.2	146	**29,7**
Копипаст %	8,1	10,7	27,4	0	
строк	739	1005	1251	0	
блоков	36	79	73	0	
файлов	10	22	6	0	

Труд	Срок	Исп		a	b	c	d
57	12	5	однородный	2,4	1,05	2,5	0,38
88	12	7	разнородный	3	1,12	2,5	0,35
135	12	11	встроенный	3,6	1,2	2,5	0,32
ч/мес	мес	чел					

Оценка по нормативным затратам (PERT). Трудоемкости по типам функциональных элементов - приблизительные

Затраты чел/час	кол-во	P	M	O	Ei	CKOi	E	CKO
Приложений	5	30	20	10	20,0	3,33	100	17
Экранов (форм)	30	10	6	4	6,3	1,00	190	30
Элем. Управления	240	5	3	1	3,0	0,67	720	160
Отчетов	12	20	10	4	10,7	2,67	128	32
Событий	130	5	3	1	3,0	0,67	390	87
Таблиц БД	20	12	5	3	5,8	1,50	117	30
Полей БД	110	3	2	1	2,0	0,33	220	37
Бизнес-объектов	25	15	5	3	6,3	2,00	158	50
Форматов файлов	5	10	8	5	7,8	0,83	39	4
						Итого	**2062**	**200**
						E 95%	**2463**	
						E ч/мес	**60**	
						Срок	**9,8**	
						Исп	**6,1**	

Управление проектом. Формирование команды. Расписание проекта.

Комментарии к документу. Для формирования команды необходимо получить примерное распределение трудоемкости и сроков исполнения проекта по этапам (фазам) и технологическим дисциплинам. Это можно сделать при помощи методики COCOMO-2 (“калькулятор” трудоемкости можно скачать http://csse.usc.edu/csse/research/COCOMOII/cocomo_downloads.htm). Хотя проект содержит много приложений, все они вписываются в единую схему клиенты-сервер и поэтому просчитываются как один модуль. Полученные данные следует считать ориентировочными (рамочными). На их основе необходимо определить функции участников проекта (с учетом совмещения) и наиболее существенные элементы **базового расписания проекта**

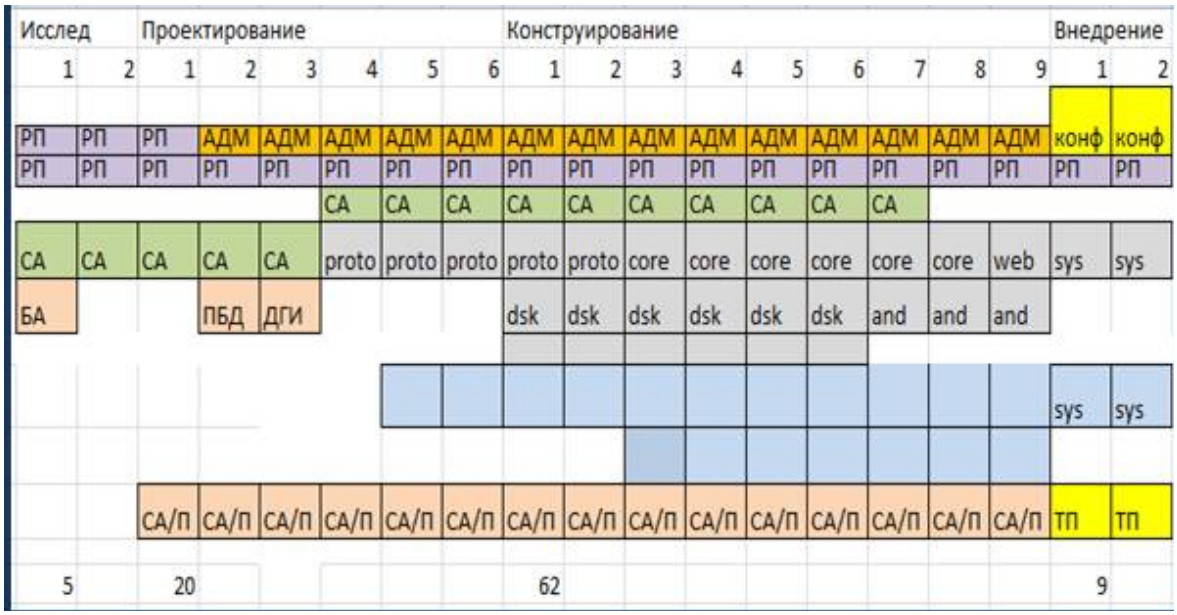
SLOC	20530			
Час/нед	160			
Ранний дизайн				
	Труд. ч-м	Срок мес	Произв. Строк/ч-м	Исп
Оптимист	54,7	13,1	375	4,2
Средний	81,6	14,9	252	5,5
Пессимист	122,4	16,9	168	7,2

Архитектурный дизайн				
	Труд. ч-м	Срок мес	Произв. Строк/ч-м	Исп
Оптимист	65,3	13,9	314	4,7
Средний	81,6	14,9	252	5,5
Пессимист	102	16	201	6,4

					Требов		Анализ		Кодир		Тестир		Управл		Окружени		Развертыв		
Исследование	4,9	1,9		2,6	1,86	1	0,93	0,5	0,39	0,21	0,39	0,21	0,69	0,37	0,49	0,26	0,15	0,08	4,90
Проектирование	19,6	5,6		3,5	3,52	0,6	7,05	1,3	2,55	0,46	1,96	0,35	2,3	0,42	1,57	0,28	0,59	0,11	19,58
Конструирование	62,0	9,3		6,7	4,96	0,5	9,92	1,1	21,1	2,27	14,9	1,6	6,2	0,67	3,1	0,33	1,86	0,2	62,00
Внедрение	9,8	1,9		5,3	0,39	0,2	0,39	0,2	1,86	1	2,35	1,26	1,4	0,74	0,49	0,26	2,94	1,58	9,79
					10,7		18,3		25,9		19,6		11		5,64		5,53		96,26
					ч-м	исп													

Кодирование	26
Тестирование	20
Анализ и проект	18
Требования	10
Управление	11
Среда разработки	7
Развертывание	4
	96

Общий список функций ролей типового проекта:
1. Группа анализа
1.1. Бизнес-аналитик
1.2. Бизнес-архитектор
1.3. Системный аналитик
1.4. Специалист по требованиям
1.5. Менеджер продукта (функциональный заказчик)
2. Группа управления:
2.1. Руководитель проекта
2.2. Куратор проекта
2.3. Системный архитектор
2.4. Руководитель группы тестирования
2.5. Отв. за управление изменениями, конфигурациями, за сборку и поставку
3. Производственная группа:
3.1. Проектировщик
3.2. Проектировщик базы данных
3.3. Проектировщик GUI
3.4. Разработчик
4. Группа тестирования:
4.1. Проектировщик тестов
4.2. Тестировщик
5. Группа обеспечения (несколько проектов)
5.1. Технический писатель
5.2. Переводчик
5.3. Дизайнер графического интерфейса
5.4. Разработчик учебных курсов, тренер
5.5. Участник рецензирования
5.6. Продажи и маркетинг
5.7. Системный администратор
5.8. Технолог



1.1. Бизнес-аналитик (БА)
1.2. Бизнес-архитектор (БА)
1.3. Системный аналитик (СА)
1.4. Специалист по требованиям
1.5. Менеджер продукта
2.1. Руководитель проекта (РП)
2.2. Куратор проекта
2.3. Системный архитектор (СА)
2.4. Руководитель группы тестирования
2.5. Отв. за управление изменениями (АДМ)
конфигурациями, сборку, поставку (конф)
3.1. Проектировщик (П)
3.2. Проектировщик базы данных (ПБД)
3.3. Проектировщик GUI
3.4. Разработчик
4.1. Проектировщик тестов
4.2. Тестировщик
5.1. Технический писатель (ТП)
5.2. Переводчик
5.3. Дизайнер графического интерфейса (ДГИ)
5.4. Разработчик учебных курсов, тренер
5.5. Участник рецензирования
5.6. Продажи и маркетинг
5.7. Системный администратор (АДМ)
5.8. Технолог
5.8. Специалист по инстр. средствам

Отдельно по исполнителям и их функциям:

1. Руководитель проекта (РП) - совмещение (частичная занятость - 0.5), полная занятость на фазе исследования: *административное управление, контроль, внешнее взаимодействие, подбор и управление командой*
2. Системный аналитик, специалист по требованиям (СА). Полная занятость на этапе исследования и первой фазы проектирования, далее частичная: *взаимодействие с пользователями, извлечение требований, формализация и постановка задач программистам*
3. Системный администратор, спец. по инструментальным средствам, отв. за управление изменениями, конфигурациями, за сборку, поставку (АДМ, конф). Частичная на этапах проектирования и конструирования, полная на этапе внедрения: *администрирование сервера БД, отслеживание исполнения задач, сборка версий, настройка конфигураций*
4. Системный архитектор, проектировщик (СА/П), этапы проектирования и конструирования: *архитектура, структура кода, программные интерфейсы, критические участки кода, шаблоны проектирования*
5. Ведущий программист (back-end разработчик): *прототип системы (proto), ядро (core), серверная часть тонкого клиента (web), бета-тестирование (sys)*
6. Программист (front-end разработчик): клиентские приложения (desktop, android) - этап конструирования
7. Разработчик тестов - этап проектирования (окончание), конструирования и внедрения: *разработка всех видов тестов и сценариев, системное тестирование*
8. Тестировщик - этап конструирования, *модульное тестирование, регрессионное тестирование сборок, тестирование графического интерфейса*, *тестирование производительности (нагрузочное)*
9. Технический писатель (ТП) - этап внедрения: *разработка документации и содержания справочной системы*

Эпизодические работы, внешние исполнители (месяцы):

1. Бизнес-аналитик, бизнес-архитектор (БА,1) - фаза исследования: *анализ предметной области, разработка видения проекта*
2. Дизайнер графического интерфейса (ДГИ,1) - фаза проектирования
3. Проектировщик базы данных (ПБД,1) - фаза проектирования (необязателен)