ToDo+Логика

28 февраля 2018 г.

11:15

1. Передавать на сервер информацию о:

Выбранном проекте (проектах)

Выбранном временнОм периоде

1. Формировать на сервере список задач, удовлетворяющих запрошенным критериям
2. Выгружать в js задачи из списка посредством json
3. При выгрузке записывать следующую информацию о задачах:
   1. Задача заблокирована от изменений пользователем с ID
4. При изменении задачи на стороне клиента отправлять изменения на сервер:
   1. Дату начала (если изменилась)
   2. Продолжительность (если изменилась)
   3. Исполнителя (если изменился)
   4. Хардлинк
   5. Софтлинк
5. Объект задача:
   1. Свойства:
      1. ID - идентификатор в глобальной базе данных
      2. Pixi\_id - идентификатор графического объекта (указатель на соответствующий графический объект)
      3. Name - краткое название задачи
      4. Text - описание задачи, что, где, когда сделать и т.д.
      5. Start - время начала задачи
         1. isLocked - может ли пересчитываться время старта при расчётах
      6. Duration - длительность задачи (для запущенных задач свойство установлено в true)
         1. isLocked - может ли пересчитываться длительность при расчётах (для завершённых задач оба свойства установлены в true)
      7. JPU - исполнитель - ссылка на уникальный ID исполнителя из другой таблицы
      8. SL - мягкие связи (softLink), последовательность задач в пределах одного конкретного исполнителя
         1. Prev - задача предшественник
         2. Next - задача последователь
      9. HL - жёсткие связи (hardLink), технологическая последовательность задач, которая не может быть нарушена
         1. Prev - задача предшественник
         2. Next - задача последователь
      10. Class - класс задачи
   2. Методы:
      1. Сдвинуть задачу вперёд: передать на вход предшественника, которому требуется сдвиг и время его планируемого завершения; рассчитать новое время старта (если требуется сдвиг); возвращать новое время завершения

**if**(endPrevJob>curNodeStart){//нужно ли сдвигать узел?

**Var** delta=endPrevJob-curNodeStart;//если нужно, то на сколько?

mxStart[curNode]=endPrevJob;//переписать новое время старта в mxStart

1. Объект исполнитель:
   1. Свойства:
      1. ID - идентификатор исполнителя в глобальной базе данных
      2. JobClasses - классы работ, с которыми может справиться исполнитель

1. Hittest на 3-х точках:
   1. Если значения в точке 1 и точке 3 равны, тогда проверить, существует ли значение в точке 1
      1. Если существует, то prevJobObj=значению из точки 1
         1. Дата старта задачи=дате окончания prevJobObj
      2. Если не существует, тогда искать возможного исполнителя, запомнить его.
         1. Дата старта задачи=точке, в которой задачу бросили
   2. Если значения в точке 1 и 3 не равны, тогда проверить, существует ли значение в точке 2
      1. Если существует, то prevJobObj=значению из точки 2, перейти к [п.2](onenote:#ToDo+Логика&section-id={B05B7FA6-2687-4CAD-80B2-F23373335EDE}&page-id={0F4DD587-A749-47C8-85DE-1C48D865099D}&object-id={B7A47C0E-5364-4F9B-B8FA-03B4967D9975}&44&base-path=https://d.docs.live.net/e342a3b17377e45f/Документы/Записная%20книжка%20пользователя%20Арсений/Что%20делать%202.one)
      2. Если не существует, тогда искать возможного исполнителя, запомнить его.
2. Вводим ограничения на возможное начало старта текущей задачи с учётом того, чтобы она не начиналась раньше предшественника из хардлинк и не завершалась до старта последователя из хардлинк, если задача выходит за границы, устанавливаем принудительно дату старта к ближайшей границе, включаем режим перебора задач исполнителя.
   1. StartCurJob определена
   2. Если имело место быть перезапись даты старта, тогда устанавливаем i=-1, чтобы считать соседей по общему алгоритму
3. Перебор задач текущего исполнителя в поисках потенциальных соседей слева и справа:
   1. Установить prevJob в -Infinity
   2. Установить nextJob в Infinity
   3. Если задача была отпущена, где попало i=-1, то:
      1. Для каждой задачи исполнителя проверяем, не является ли она той самой текущей задачей, которую перетаскивают:
         1. Если является, то: пропустить её, перейти к следующей, пока не закончатся
         2. Если не является, то:
            1. Ищем соседа слева:

Определяем дистанцию между датой старта текущей задачи и датой окончания перебираемой,

Для случая, если она положительна, сравниваем её с предыдущей обнаруженной минимальной дистанцией:

Если текущая дистанция меньше сохранённой минимальной, то:

prevJob=перебираемая задача

Сохранённая минимальная дистанция=текущей дистанции

Если дистанция отрицательна, тогда ищем соседа справа:

Определяем минимальную дистанцию между датами старта перебираемой и текущей задач

задача, для которой дистанция будет минимальна - будет соседом справа - nextJob

* 1. Если задача была отпущена на другой задаче и i ≠ -1:
     1. prevJob = prevJobObj.jobID
     2. Если prevJobObj.jobID
        1. совпадает с iiCurJobObj.prev, значит задачи не поменяли своего взаимного расположения после перетаскивания, и nextJob=iiCurJobObj.next
        2. Не совпадает с iiCurJobObj.prev, значит взаимное расположение задач изменилось и nextJob=prevJob.next

1. Обработка бесконечностей:
   1. nextJob- задача, которая станет следующей для текущей после её переноса
   2. prevJob - задача, которая станет предшественником для текущей после её переноса
   3. prevCurJob - задача-предшественник текущей до начала переноса
   4. nextCurJob - задача-последователь текущей до начала переноса
   5. iiCurJobObj - объект текущей задачи, которую переносят
   6. Исключения:
      1. prevJob =-Infinity, если задача была перенесена в самое начало
      2. nextJob = Infinity, если задача была перенесена в самый конец
2. Запись информации о новых связях задач в софтлинк:
   1. Склеить место разрыва:
      1. В prevCurJob.next записать ссылку на nextCurJob
      2. В nextCurJob.prev записать ссылку на prevCurJob
   2. Сформировать новые связи с новым предшественником:
      1. В prevJob.next записать iiCurJobObj.jobID
      2. В iiCurJobObj.prev записать prevJob
   3. Сформировать новые связи с новым последователем:
      1. В iiCurJobObj.next записать nextJob
      2. В nextJob.prev записать iiCurJobObj.jobID

При отпускании перетаскиваемой задачи проверять:

1. Если строка больше максимальной (т.е. задачу отпустили в ведре)
   1. Если задача из основного контейнера с задачами, значит её хотят перенести в ведро, в таком случае надо:
      1. Если у задачи нет ни предшественников, ни последователей, то:
      2. Стереть исполнителя из задачи
      3. У исполнителя стереть задачу
      4. Определить конкретную строку внутри ведра
      5. Опустить все задачи, начиная с этой строки, на 1 строку ниже
      6. Поместить текущую задачу на освободившуюся строку
      7. Определить последователей из хардлинка
      8. Перенести последователей из хардлинка на строку, следующую за текущей (или хотя бы в новые последние строки)
   2. Если задача из ведра, значит её хотят перенести внутри ведра на другую строку, в таком случае надо:
      1. Определить конкретную строку внутри ведра
      2. Опустить все задачи, начиная с этой строки, на 1 строку ниже
      3. Поместить текущую задачу на освободившуюся строку
2. Если строка равна какой-то из разрешённых:
   1. Если задача из основного контейнера, значит совершить обычный перенос
   2. Если задача из ведра, значит:
      1. Проверить, есть ли у задачи предшественники (и предшественники предшественников и т.д.)
         1. Выводить сообщение вида "следующие задачи будут автоматически добавлены в план" - да/нет.
            1. Добавлять задачи, согласно классам/исполнителям
            2. Если указан конкретный исполнитель, добавлять задачу ему после последней имеющейся
            3. Если указан класс, добавлять задачу тому исполнителю класса, который освобождается раньше всех.
      2. Записать в задачу исполнителя и время старта, обновить ссылку на контейнер pixi.

Если в плане есть задача, последователи которой в хардлинке в ведре, то отмечать эту задачу отдельным цветом, каким-то бледным.

Связи в хардлинк можно строить в ведре!

Задачи в ведре расположены в цепочках. ~~Одна строка - одна цепочка~~.

Календарь можно реализовать несдвигаемыми задачами-выходными-отпусками-больничными.

При создании объектов задач проверять, существует ли исполнитель с указанным ID, если нет, то создавать его; дописывать ему задачи в массив задач.

Ortems и quintiq Dassault

1. Интерфейс: должно быть дерево проектов
2. Каждая задача принадлежит определённому проекту;
   1. не бывает ничейных задач
   2. При создании задачи, ей автоматически прописывается в хардлинк первая и последняя задачи проекта
   3. При создании задачи, она имеет:
      1. Длительность 1 день
      2. Время старта рассчитывается таким образом, чтобы она завершалась непосредственно перед наиболее ранним последователем в хардлинк
      3. Исполнитель по-умолчанию - ведро соответствующего класса, либо общее ведро проекта
      4. Вызывается функция отрисовки созданной задачи со сдвигом всех последующих задач