Задание AtomSkills 2025 — Программные решения для бизнеса

Краткое описание

Компания «Definitely Not a Galley Inc.» имеет собственный штат разработчиков и других IT-специалистов, которые реализуют разработку под ключ различным заказчикам. Благодаря удачной рыночной конъюнктуре в последние годы и самоотверженному труду персонала, компания расширялась взрывными темпами. Только за 2024 год объем заказов и штат сотрудников выросли многократно. Для того, чтобы снизить операционную нагрузку, в компании решили провести автоматизацию некоторых процессов, которые с момента образования компании выполнялись вручную. Разработку информационной системы для этой автоматизации в компании решили отдать на аутсорс.

Компания «Definitely Not a Galley Inc.» позиционирует себя на рынке как лидер инноваций. Кроме того, в компании гордятся тем, что могут решить задачу любой сложности за разумное время. Поэтому от разработчиков будущей системы ожидается решение алгоритмически сложной задачи лучшим из возможных способов.

Контекст

Компания «Definitely Not a Galley Inc.» имеет собственный штат разработчиков и других IT-специалистов, которые реализуют разработку под ключ различным заказчикам. Заказчик присылает требования, а аналитики компании разбивают их на задачи от старта до финиша. После этого компания должна распределить задачи по всем своим специалистам наиболее эффективным образом. До сих пор она делала это вручную, но после найма большого количества новых специалистов пришла к выводу, что этот процесс нужно автоматизировать.

Термины и определения

Заказ — определённый набор *задач*, для которых указано, какие из этих задач зависят друг от друга, а какие могут быть сделаны параллельно. Ещё у каждого заказа обязательно есть *крайний срок сдачи*, *стоимость* и *штраф за просрочку*.

> **Крайний срок сдачи заказа** — момент во времени, к которому должны быть закончены ВСЕ задачи заказа.
>
> **Стоимость заказа** — денежное вознаграждение, которое компания получает за своевременное выполнение заказа.

Штраф за просрочку в день — сумма, на которую уменьшается стоимость заказа при просрочке заказа на *календарный* день. Итоговая стоимость заказа не может быть меньше нуля.

Задача — некоторая операция, которая может в один момент времени выполняться только одним *работником*. У задачи есть *тип* и *базовая длительность*.

Тип задачи — значение перечисления с типом выполняемой работы, например, архитектура, дизайн, разработка на определённом языке, деплой.

Базовая длительность задачи — время в рабочих днях, которое требуется *работнику* с *уровнем производительности 1*, чтобы выполнить эту задачу.

Работник — разработчик, дизайнер, архитектор, девопс или тестировщик. У работника есть *список типов задач*, которые он может делать, а также *уровень производительности*, показывающий опыт и скорость работы.

Уровень производительности работника — число от 0.5 до 2.0, тем большее, чем быстрее этот работник может выполнять задачи своих типов. Опытные и компетентные работники делают задачи быстрее с тем же уровнем качества. Итоговая длительность задачи всегда в целых днях с округлением в большую сторону (то есть, если работник утром взял какую-то задачу, то другую в этот день он уже не возьмёт).

Общие данные системы

В системе присутствует набор общих данных, формат которых установлен в приложении. Содержимое общих данных:

Список типов задач — все типы задач, которые могут выполнять работники данной организации.

Список работников — все нынешние сотрудники организации.

Текущая дата — какую дату считать текущей на момент начала работы системы.

Стоимость дня работы компании — каждый *календарный* день работы компании стоит определённую сумму денег. Сумма не зависит от того, насколько нагружены работники, потому что все они получают обычную фиксированную зарплату, в том числе если уходят на простой (см. ниже). Фирма тратит деньги даже в выходные и праздники.

Список праздничных дат — в праздничные даты и в выходные работники не делают задачи, но фирма всё равно тратит деньги. Все дедлайны и длительности выполнения задач считаются только по рабочим дням.

Функциональные требования

Функция загрузки общих данных

В интерфейсе системы должна быть опция загрузки общих данных посредством передачи в систему JSON с установленной в приложении структурой.

Также должна быть возможность, не перезапуская систему, загрузить в неё повторно новый набор общих данных с перезаписью. Новые общие данные полностью перезаписывают старые. Загруженные заказы и построенный план работ при этом стираются.

Добавление заказов

В системе должен быть способ пакетно загрузить в неё заказы, описанные в JSON-файле в формате, представленном в приложении. Типы работ в заказах опираются на те, которые были переданы системе во входных данных. Одновременно в системе может находиться до 500 заказов и до 5000 задач в общей сложности.

Операцию загрузки заказов можно выполнять несколько раз, при этом новые должны добавляться к старым с исключением дубликатов. Заказы и задачи с одинаковыми идентификаторами считаются полностью одинаковыми, редактировать их содержимое не нужно. Новые заказы с дедлайном в прошлом не нужно присоединять к системе, но следует уведомить пользователя о том, почему они не были загружены.

Очистка заказов и плана работ

В системе должна быть возможность очистить систему от старых данных, удалить одним действием все заказы и построенный план работ.

Установка текущей даты

Система должна позволять в любой момент передвинуть текущую дату на один или более дней вперёд (от прошлого к будущему). Первоначальная «текущая» дата задаётся при загрузке первичных данных. При установке даты берётся в расчёт только год, месяц и день.

Планирование работ

Система должна содержать функцию, позволяющую автоматически распределить задачи во времени и по работникам так, чтобы **максимизировать общую прибыль**. Часть заказов могут быть исключены, если с точки зрения максимизации прибыли это оправдано.

Распределение заказов должно подчиняться следующим требованиям:

- На работника не может быть назначена задача того типа, который не перечислен в списке типов работ, выполняемых этим работником
- Работник не может выполнять в один момент времени более одной задачи
- Работник не может выполнять задачи, если он простаивает (см. ниже «Простой работника»)
- Работник не работает в выходные (суббота, воскресенье) и праздничные (задано в исходных данных) дни
- Для части задач установлена зависимость друг от друга: зависимая задача должна быть начата позже, чем закончатся все задачи, от которых она зависит
- Заказ считается выполненным только тогда, когда выполнены все его задачи; если дедлайн заказа больше или равен дню окончания последней его задачи, то заказ выполнен без штрафа

При любом изменении данных или при добавлении новых заказов должна быть возможность запустить пересчёт плана, но только для периода времени, начинающегося с «текущей даты».

При оценке решений будут загружены несколько наборов тестовых данных, отличающихся от тех, которые выданы командам для разработки. Для этих данных заранее предусмотрена шкала градации суммарного достигнутого алгоритмом дохода за все выполненные заказы.

Общая прибыль по всем заказам рассчитывается следующим образом:

1. По каждому выполненному заказу считается его стоимость, суммируется
2. Если заказ выполнен, но просрочен, из стоимости вычитается штраф за каждый календарный день просрочки, при этом результирующая стоимость заказа не может быть отрицательной, но может быть нулевой; повторяется для всех заказов
3. За каждый календарный день вычитается стоимость дня работы фирмы независимо от загруженности работников и распределения задач между ними

Временем работы фирмы считается число дней между датой начала самой первой задачи и датой окончания самой последней, включительно.

Отображение данных

Система должна предоставлять пользователю возможность просматривать информацию по загруженным данным: общее число работников, заказов, задач.

В любой момент времени система должна отображать установленную текущую дату (не обязательно равную фактический дате запуска приложения).

Требуется отображение текущего статуса пересчёта. Если система занята расчётом, требуется индикация этого процесса.

Также интерфейс системы должен позволять увидеть:

1. Метаданные по каждому работнику
2. План по каждому работнику: какую задачу он выполняет в любой момент времени, либо простой, если он простаивает
3. Метаданные каждого заказа и задач в нём
4. План по каждому заказу: какие задачи этого заказа, какими работниками и когда будут выполнены
5. Время завершения каждого заказа, успевает ли заказ к крайнему сроку, а если он просрочен, то на сколько дней
6. Заказы, полностью исключённые из сетки планирования, если такие имеются
7. Сумму прибыли по каждому заказу с учётом штрафа
8. Общую сумму прибыли за все заказы с учётом расходов на дни работы фирмы

Печать плана для работника

Интерфейс системы должен позволять вывести на печать план работ для любого выбранного работника с описанием того, какие задачи, для каких заказов и в какие моменты времени он должен делать, а также информацию о заданных простоях этого работника.

Время на оптимизацию

Гарантируется, что в систему не будет загружено больше, чем:

- 5000 задач одновременно
- 500 заказов одновременно
- 100 работников одновременно

С учётом этих показателей, допустимое предельное время оптимизации плана работ:

- В худшем случае: не более 300 секунд
- В среднем: не более 60 секунд

Изменение состояния задачи

Если задача начинается не позже, чем «текущая дата» (установленная в системе), и заканчивается не раньше, чем «текущая дата», то должна быть возможность задать вручную или отредактировать заданные прежде:

- Реальный процент выполнения задачи на текущий момент
- **ИЛИ** оставшуюся длительность выполнения задачи на текущий момент

Задание любого из этих двух показателей автоматически пересчитывает и определяет второй показатель.

Момент окончания работы над задачей не может быть меньше текущей даты.

Простой работника

В интерфейсе системы должна быть возможность любому работнику задать период недоступности с любой даты, не раньше текущей, до любой другой более поздней заданной даты.

Выгрузка плана работ

В системе должна быть возможность выгрузить общий план работ в виде JSON-файла в машиночитаемом виде. Формат этого JSON приведён в приложении. Система должна всегда выгружать текущий план работ, который видит пользователь в интерфейсе в данный момент.

Документирование системы

Для передачи системы потенциальному заказчику необходимо разработать следующий комплект документации:

1. **Инструкция по развертыванию** — именно этой инструкцией будет пользоваться системный администратор при установке системы. Если он не сможет развернуть систему по данному документу, то заказчик не примет такой проект к использованию.
2. **Техническая документация** — описание архитектуры; разнообразные схемы и документы по проекту для быстрого понимания: что, где и зачем в проекте лежит, по каким правилам работает, включая структурную схему БД или иного источника данных.
3. **Математическое описание** — описание использованных алгоритмов, применённых подходов для оптимизации, математические и статистические оценки, которые делала команда в процессе разработки.
4. **Инструкция пользователя** — описание того, как именно задействовать функциональность системы. Человек, не знакомый с программой, должен иметь

возможность по инструкции достичь результата, описанного в функциональных требованиях.

5. **Руководство администратора** — описание того, как конфигурировать систему после развёртывания. Также в данной инструкции должны быть указаны реквизиты или способ для первичного входа и администрирования приложения, если таковые требуются.
6. **Исходный код** программного продукта.

Пожелания клиентов

Клиенты высказали ряд пожеланий, которые следует учесть при разработке системы:

- Разрешено использовать только Open Source компоненты
- В случае возникновения ошибок в процессе использования, система должна выдавать информативные и понятные пользователю сообщения об ошибках
- Интерфейс должен быть эстетически приятным, понятным, использовать стили и подходы современного проектирования программных продуктов

Представление проекта заказчику

После разработки продукта, его следует представить заказчику. Помимо очевидного выделения финансов, заказчик при окончательном выборе подрядчика будет смотреть на следующие вещи:

- Общее описание реализованной системы: какие у неё есть функции и отличительные черты
- Оценка стоимости разработки системы; какие специалисты потребовались для разработки, какие будут нужны для поддержки
- Описание технологий, использованных в проекте: почему были выбраны эти технологии, какие у них плюсы и минусы
- Описание выбранного алгоритма, оценка скорости работы и объёмов данных, с которыми система может работать; оценка качества оптимизации
- Возможные пути развития системы, предложения по улучшению

Приложение

Уточнение по датам

Для периодов времени следует учитывать следующие требования:

Параметр	**Откуда берётся**	**По каким дням**
Базовая длительность задачи	Исходный параметр задачи	По рабочим дням, исключая выходные и праздники
Итоговая длительность задачи	Базовая длительность после пересчёта по производительности работника с округлением вверх	По рабочим дням, исключая выходные и праздники
Просрочка выполнения заказа	Разница между датой готовности заказа и датой дедлайна, в целых днях	По календарным дням
Общее время работы фирмы	Разница между датой окончания последней задачи и датой начала первой, в целых днях, включительно	По календарным дням

Входные данные

Единый JSON-файл с пятью верхнеуровневыми полями: список типов работ, список сотрудников, текущая дата, стоимость работы фирмы в день и список праздничных дат.

```
{
	"workTypes": [ // список существующих типов работ
		{
			"name": "Разработка на C#", // название типа работ
			"id": "DevCSharp" // идентификатор типа работ
		},
		...
	],
	"companyDayCost": 12500,	//	стоимость дня работы всей фирмы
	"currentDate": "2025-08-09", //	дата, которая считается
					текущей при запуске программы
	"holidays": [ "2025-05-01", "2025-12-31", "..." ] // праздничные даты
	"workers": [ // список работников
		{
			"id": "IvanPetrov", // уникальный идентификатор работника
			"name": "Иван Петров", // имя работника
			"workTypeIds": [ //	список идентификаторов
					типов работ данного работника
				"DevCSharp",
				"DevJava",
				"..."
			],
			"productivity": 1.3 // производительность
		}
	]
}
```

Добавление заказов

Импортироваться будет большой JSON со списком заказов

```
[ // список заказов
      { // объект одного заказа
            "id": "123ABC", // уникальный идентификатор заказа
            "tasks": [ // список задач этого заказа в произвольном порядке
                  {
                        "id": "123ABC_456DEF", //      уникальный
                                                      идентификатор задачи
                        "workTypeId": "Design", //     идентификатор
                                                      типа работы в этой
                                                      задаче
                        "dependsOn": [ // список идентификаторов задач
                                          этого же заказа,
                                          от которых зависит текущая задача
                              "123ABC_789GHI",
                              "123ABC_012JKL",
                              "123ABC_345MNO",
                              "..."
                        ],
                        "baseDuration": 5 // базовая длительность задачи
                  }
            ],
            "deadline": "2025-08-11", //   в конце этого дня
                                          заказ должен быть выполнен
            "earning": 5230000, //         заработок с заказа
                                          при своевременном выполнении
            "penaltyByDay": 5230 //        штраф за каждый день просрочки
      },
      ...
]
```

Выгрузка плана

JSON-массив с задачами, выстроенными во времени и распределёнными по работникам:

```
[ // список всех задач
      { // объект одной задачи после распределения
            "taskId": "123ABC_789GHI", //  идентификатор задачи,
                                          в точности равный её
                                          идентификатору во входных данных
            "workerId": "IvanPetrov", //   идентификатор работника,
                                          на которого эта задача назначена
            "start": "2025-08-11", //      в начале этого дня задача
                                          запущена в работу
            "end": "2025-08-15", //        в конце этого дня задача
                                          будет завершена
      },
      ...
]
```

Для обозначения простоя нужно ввести в идентификатор задачи `null` или пустую строку.