Техническое задание

Разработать модуль логирования для сайта на Laravel

Общее

Есть базовое приложение на Laravel (компании, пользователи, статьи, товары).
Нужно сделать модуль, который будет логировать изменения с объектами, связывать логи с самими объектами, выводить логи на фронтенд в нескольких вариантах отображения.

Для выполнения задачи предоставляется доступ к базовому (тестовому) приложению. После сдачи работ результаты будут перенесены в рабочее приложение силами заказчика.

Описание логики базового приложения

Есть компании (организации). В компаниях пользователи.
Пользователи авторизуются и им становятся доступны объекты компании.
Объекты есть 2 видов: статьи (posts) и продукты (products). Связь между ними многие к многим.
Объекты могут быть в 2 состояниях - приватные (доступны только конкретной компании) и публичные (доступны всем компаниям и неавторизованным пользователям).
Пользователи могут создавать/изменять/удалять объекты своей компании. Пользователи могут публиковать объекты своей компании или добавлять в свою компанию публичные объекты, при этих операциях создаются новые экземпляры объектов и между ними устанавливается связь.
Список всех компаний и пользователей на главной странице, пароль у всех password.

Технические характеристики

- Фреймворк Laravel
- Верстка bootstrap
- Используемые JS компоненты на фронтенде jquery, bootstrap, datatables, select2, daterangepicker, moment.js, pnotify
- Используемые пакеты composer: laravel-datatables, livewire

Приложение развернуто на VDS, опубликовано в интернет.
Будут предоставлены доступ по ssh и к phpmyadmin.
Приложение хранится в приватном репозитории github, настроена связь с экземпляром на VDS.
Будет предоставлен доступ к репозиторию для вашей учетной записи github (для заливки изменений и результатов работ).
Работу можно вести как на предоставленном VDS так и разворачивать приложение у себя, как удобнее.
Установка и использование для работы дополнительных компонентов – по согласованию.

Приемка работ

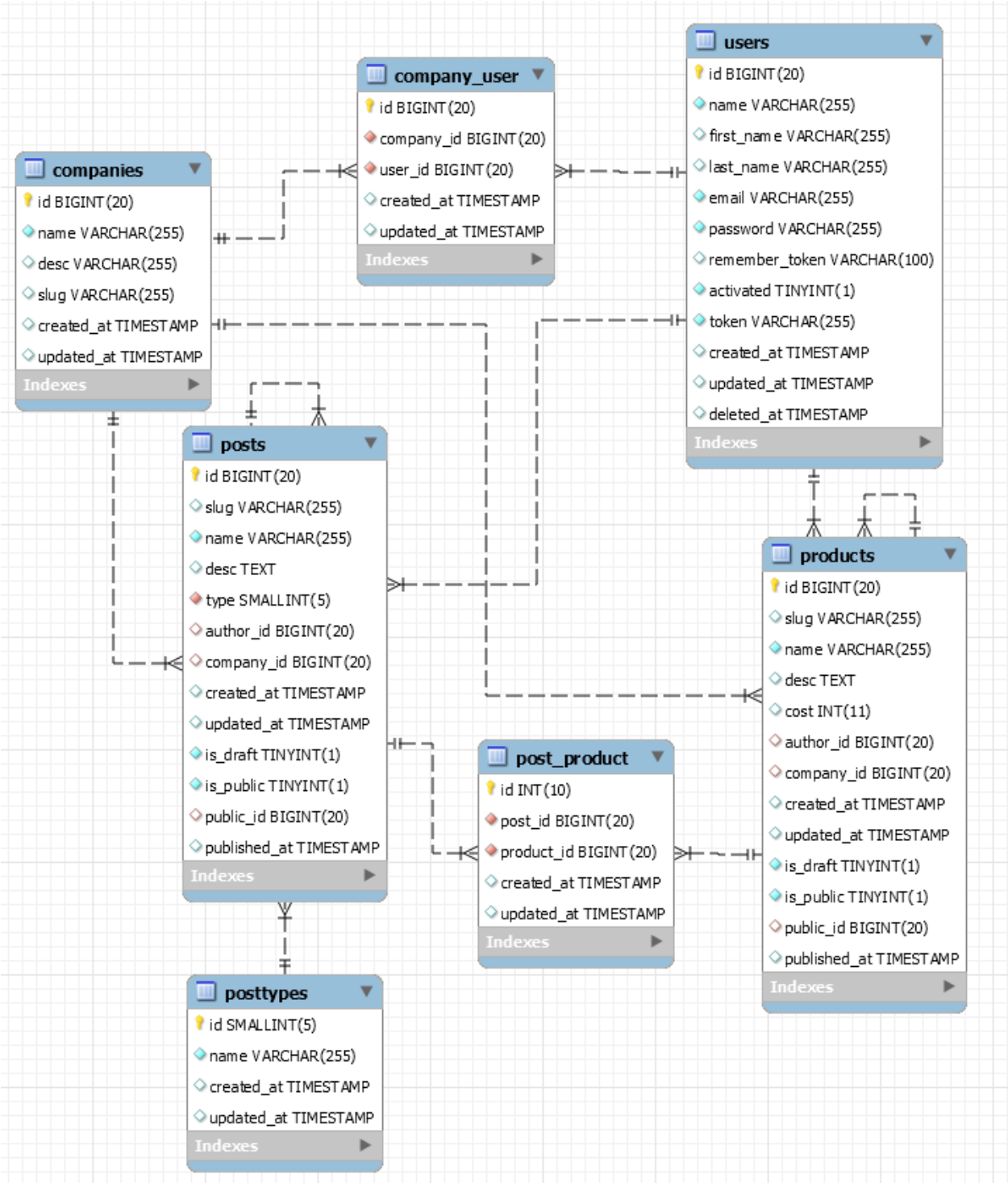
По результатам работ необходимо:

- Продемонстрировать работоспособность (выполнение ТЗ) на предоставленном VDS сервере заказчика.
- Залить все изменения на Github. Не перезаписать репозиторий, а закоммитить, чтобы было видно, какие именно изменения были внесены в код.

Оплата

Как вам удобнее

Схема БД



(служебные таблицы не приведены)

Задание

Логированию подлежат операции, совершаемые пользователями или системой: создание, изменение, удаление любых видов объектов.
Логи должны связываться с объектами (одним или несколькими, разных типов).
На сайте могут быть разные объекты (=таблицы) - news, articles, products, groups и т.п., список не конечен. Модуль должен быть универсален и работать с множеством разных объектов.
У каждого события может быть 1 источник (инициатор) - это либо авторизованный пользователь (таблица users) либо СИСТЕМА (для событий, прошедших по расписанию, без участия пользователя).
У событий обязательное поле COMPANY_ID. Если есть инициатор события то company_id = users.company_id, иначе company_id берется из объектов, изменение которых вызвало лог.

Таблица EVENTS:

- id [bigint index]
- author_id [bigint], =users.id или null если событие вызвала СИСТЕМА
- company_id [bigint], =companies.id, обязательное поле.
- type [int], = 1,2,3,... = тип события create, change, delete, могут быть и любые другие типы, например sendmail, publish
- message [string]
- old [text]
- new [text]
- created_at

Связь события с объектами

Каждое событие может быть связано с множеством иных объектов.
Например, пользователь создал Статью (posts). Мы создали событие которое связано с ID созданной статьи.
Или иной пример, установили связь статьи и продукта. Созданное событие нужно связать и с статьей и с продуктом.
Связи задаются в отдельной таблице.

Таблица EVENTS_OBJECTS:

- id [bigint]
- event_id [bigint, = events.id]
- object_type [string, = название таблицы, например users, posts, products, и т.п.]
- object_id [bigint]

Должна быть возможность к любому создаваемому событию добавлять любое количество связей с любыми объектами, что-то вроде:
$log->addConnections (['posts']=>$post, ['producs']=>$products) где передается либо конкретный объект с которым нужно установить связь либо коллекция объектов.

Поля Old/New

Эти поля нужны для того, чтобы хранить значения измененных (созданных) параметров объектов.
Для событий type=create заполняется только поле new, для type=delete поле old, для type=change заполняются и old и new.
Много изменений объекта могут произойти одновременно, чтобы не создавать по отдельному событию в логе для каждого измененного поля, в полях old/new размещается массив (json или что то иное) с парами ключ-значение, где ключ - название добавленного/измененного поля на английском а значение - текстовое значение.

Пример, создали новый пост, при создании заполнили поля name и desc. В events.old будет пусто, в events.new будет *{"name":"Новый пост","desc":"И его описание"}*
Еще пример, у поста изменили описание. В events.old будет *{"desc":"И его описание"}* а в events.new будет *{"desc":"Новое описание поста"}*
Если объект удаляют то в events.old заносится его поле name *{"name":"Название удаленного объекта"}* или то, что подадут на вход при генерации лога.
Если изменяемое поле - ссылка на другую таблицу то в events.old/new заносится не id содержащийся в измененном поле а name из соответствующей связанной таблицы.
Пример, у Поста поменяли тип с Статья на Новость. Поле posts.type, ссылается на таблицу posttypes (= posttypes.id, один к одному), при изменении поля в логе должно быть events.old = {"type":"Статья"} events.new= {"type":"Новость"}.

Если изменилась связь Многие ко Многим то в events.old записываются названия объектов с которыми связь была разорвана а в events.new названия объектов с которыми связь была добавлена.
Пример, у статьи поменяли связи с продуктами, удалили один продукт и добавили 2 новых. В логе должно быть events.old = {"connections":["products":Удаленный продукт]"} events.new= {"connections":["products":"Добавленный продукт 1";"products":"Добавленный продукт 2"]}.

Связи многие ко многим

События создания или разрыва связи объектов многие ко многим, например - связь статьи с товарами, должна быть возможность сохранять как отдельные логи. Не по умолчанию а по потребности. И этот лог должен быть связан с несколькими объектами.
Добавление связи это events.type=create, удаление связи это events.type=delete
Для каждой связи создается отдельное событие в логе.
Зачем это нужно, почему нельзя отразить операции над связями только в полях old/new ? - потому что такое событие одинаково связано и с posts и с products, соответственно отображаться должно в карточках обоих объектов.
Но этот функционал должен быть возможностью, а не по умолчанию для всех т.к. могут быть связи один ко многим когда дополнительный лог не нужен. Например, если мы решим добавить posts.lotoftypes с несколькими вариантами выбора, нам будет достаточно лога для posts, для lotoftypes создавать логи будет не нужно т.к. это не самостоятельный объект (как например products с своей страницей, списками и т.п.) а всего лишь составляющая объекта posts.

Создание событий логирования

Запуск модуля логирования (создание лога) осуществляется вызовом в соответствующих контроллерах объектов. Пример того, как могут создаваться события:
Пользователь добавил новую статью, связал ее с несколькими товарами, отработал PostController::store(), в нем после сохранения продукта примерно такой код для создания лога:

```
$log = new events([
  'type' = ''edit";
  'message' = trans('primary.events.PostWasEdited'); // = "Публикация изменена"
   'old' =>$oldPost
   'new' => $post
]);
```

Незаполненные поля формируются автоматически. Например author_id = Auth()->id или NULL если событие выполнила система. company_id = берется компания пользователя или изменяемого объекта

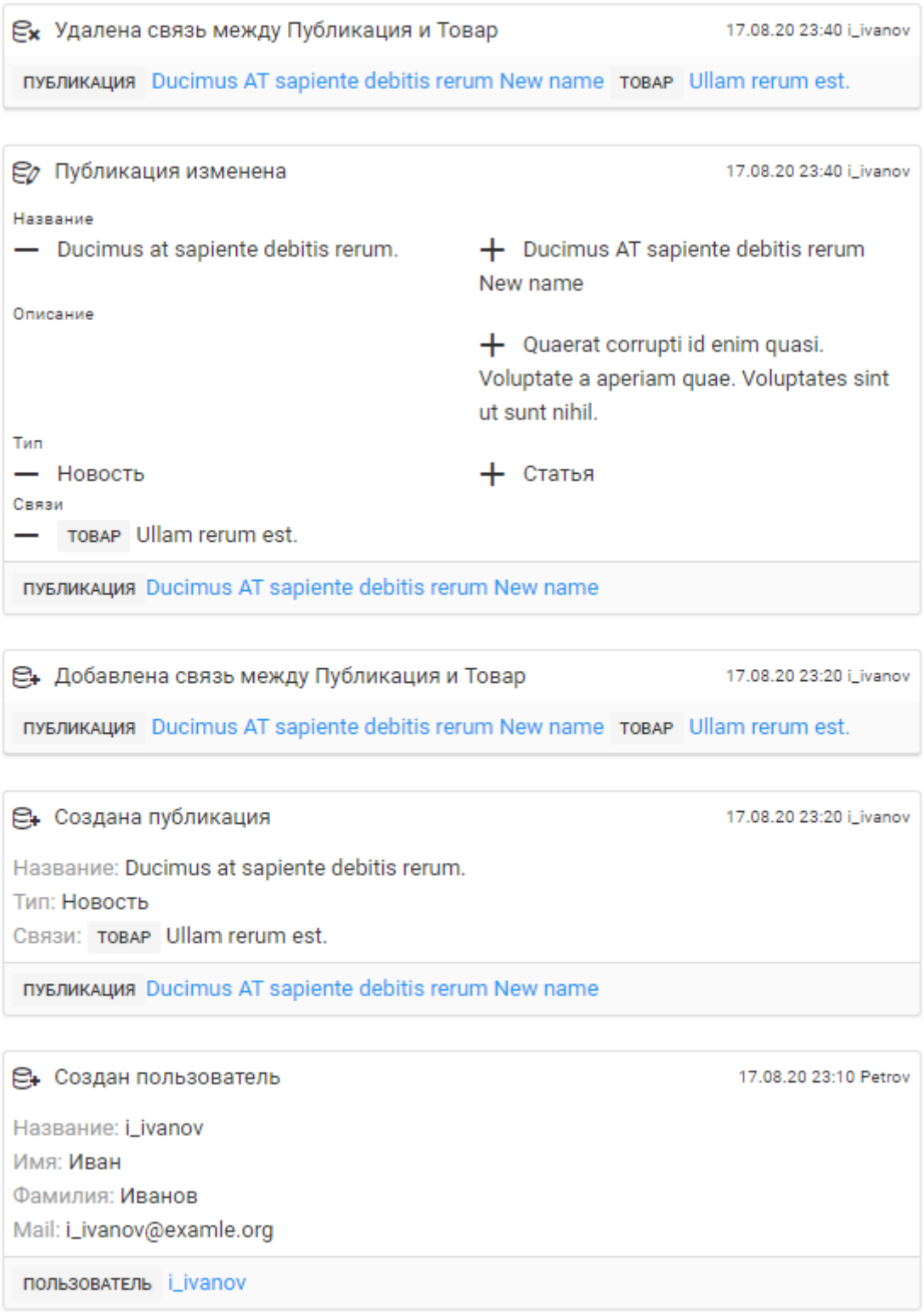
Отображение логов на сайте

Логи могут отображаться в 2х местах

1. На страницах связанных с логом объектов
2. На странице логов

Страницы связанных с логами объектов

На страницах объектов (статьи, продукты, пользователи и любые другие) нужно добавить кнопку logs. По нажатии на нее всплывает modal с всеми логами, связанными с этим объектом.
По умолчанию, при загрузке страницы, логи не загружаются. Грузятся по ajax или Livewire при наведении мыши на кнопку. Отображение:



(верстка будет предоставлена)

Страница логов

По адресу site.adress/logs отображается таблица с логами, использовать DataTables с подгрузкой данных по ajax. Пример реализован на странице /products, можно взять за основу.
К DataTables добавить кнопку с фильтрами (SearchPanes Extension). Фильтры:

- по имени пользователя, вызвавшего событие (author_id)
- по типу операции (создание, изменение, удаление)
- По виду объектов, с которыми связан лог (users, posts, products и т.д.)

К странице с логами должна быть возможность обратиться по ссылке с дополнительными параметрами. Пример:
site.adress/?author=4 - отфильтрует события у которых author_id = 4
site.adress/?type=delete - отфильтрует события у которых type=delete
site.adress/?users=4 - отфильтрует события связанные с пользователем с id=4
site.adress/?posts=4 - отфильтрует события связанные с статьей с id 4
И любые другие типы объектов (products например)
site.adress/?author=4&posts=4 - отфильтрует события связанные с статьей с id=4 которые произвел пользователь с id=4
И любая иная комбинация из указанных параметров.

Поля таблицы: Дата, Источник (author_id), Тип, Событие, Связанные типы объектов
В поле Событие на новой строчке добавляются связанные объекты в виде ссылок.

Отображение: