***Скинуть ссылки***

<https://rusfriend.com>

<https://github.com/MrCoul3/rusfriend.com>,   
<https://www.figma.com/file/G2A8muq6LKzoFAtYW9fnr4/russian-friend.COM-DIPLOM?node-id=359%3A4021>

Зарегистрировать двух пользователей

1. **Введение**

**Заказчик: преподаватель русского языка как иностранного**

**Цель сайта: Коммерческая образовательная платформа по русскому языку для иностранцев.**

**Функции сайта: сайт визитка, продажа видео курсов, бронирование индивидуальных онлайн занятий**

**Целевая аудитория: Путешественники, эмигранты, работники международных компаний, любители культуры, межнациональные браки.**

**Чувства посетителей после посещения веб-сайта: Желание приобрести продукт. Начать изучать язык не откладывая.**

**Схема сайта – картинка**

***Скинуть ссылку на сайт***

*Задачей стояло сделать сайт – платформу для онлайн школы с нуля. У заказчика не было вариантов дизайна, только примерные представления о структуре. Первым делом был составлен бриф. Определена целевая аудитория, поставлены цели сайта, какие действия будут совершать посетители на сайте (функции сайта). Что должны испытывать посетители, посетив веб-сайт. Придуманы варианты названий и логотипа.  
 Произведен анализ сайтов – конкурентов, определены плюсы и минусы, заимствованы лучшие идеи и учтены ошибки и недостатки у каждого из них.*

*Определились с отображением сайта на устройствах с различным разрешением – решили делать под все устройства, начиная с десктопной версии.*

*Составлена структура сайта – представлена на схеме. (По мере работы над сайтом схема дополнялась.) Красным показаны элементы еще не реализованные, но задуманные для внедрения.  
Сайт состоит из основной части – для пользователей – главная страница, обо мне, курсы, индивидуальные занятия, спикинг клаб и гайд и административной части – админ – панели функция которой установить расписание, подтвердить, отменить, перенести урок, получить информацию о зарегистрированных пользователях, изменить цену занятия и курсов, добавить или удалить курсы (еще не реализовано).*

1. **Дизайн**

**Первоначальные варианты дизайнов: вставить картинки (вар\_1,2,3)**

**Шрифты: Exo 2, Montserrat.   
Заголовки h1: 48px; h2: 28px; h3: 20px;**

**Цветовая палитра вставить палитру**

***Дизайн****(от*[*англ.*](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)*design — проектировать, чертить, задумать, а также проект, план, рисунок)*

*Скинуть всем ссылку на фигма  
Вся работа по макетированию проекта выполнена в программе FIGMA.*

***Слайд 6****,* ***7*** *Набросав структуру главной страницы приступил к ее макетированию. На слайдах можно увидеть первоначальные варианты дизайнов.*

***Слайд 8***

*В процессе макетирования появлялись новые идеи, придумывались разные фишки, начал проясняться окончательный образ сайта.  
В итоге решено использовать минималистичный дизайн, яркие цвета – с акцентом на красный. Минимальное количество шрифтов и размеров (2 шрифта).*

***Слайд 9*** *Определились размерами кнопок на разных экранах, их цветом и анимацией. (анимация – проблескивание – готовое решение - реализовано с помощью css-keyframes с бесконечным циклом повторения раз в три секунды common.scss 88)  
Так же нашли подходящие иконки.*

*Приняли размеры сайта для разных устройств (для медиазапросов)*

*(*Подключение метатега viewport к странице в большинстве случаях осуществляется так: <meta name="viewport" content="width=device-width, initial-scale=1">)

1. **Структура и код**

**Для создания сайта я использовал:**

Phpstorm

Figma

Webpack

OpenServer

Vue.js

JQuery

PHP

Composer

***Слайд 10*** *При создании сайта я использовал phpstorm (хоть и начинал с VSCode, понял что шторм функциональнее). Преимущества: удобно продолжать работу с файлами - в каждом открытом файле запоминается курсор и место где вы редактируете. Синхронизация проекта с удаленным сервером – ручная и автоматическая. Последняя – очень удобная функция, сильно облегчает жизнь и экономит время. Встроенный терминал, работа с GIT. Удобный поиск по любым параметрам.*

***Слайд 11***

*После макетирования я приступил к верстке. Используя webpack, сильно облегчил себе жизнь и сократил время создания html разметки и css.*

*(* Cascading Style Sheets «каскадные таблицы стилей») — формальный язык описания внешнего вида документа,  HyperText Markup Language — «язык гипертекстовой разметки») — стандартизированный язык разметки документов*)*

*На слайде 11 представлена структура каталогов*

*В верстке использовал методологию БЭМ – тк данный прием облегчает дальнейшую поддержку сайта, код становится визуально более доступным.*

***Слайд 12, 13***

*Начал с шаблона – template.pug. Его можно посмотреть открыв файл по адресу rusfriend.com/src/pages/include/template.pug на ГитХаб.*

*Шаблон состоит из метатегов , заголовка, блока Header, блока main, и блока footer. Шаблон подключается ко всем страницам сайта кроме админ панели (там свой шаблон).*

***Слайд 14****Мы видим страницу index – Главная, состоит из команды extends для покдлючения шаблона, блока констант, и блока контент который находится в блоке мэин подключаемого шаблона. Страница построена по принципу секций с помощью тега section. Все остальные страницы по такому же принципу.*

***Слайд15****Стили описаны с помощью модуля scss (*Sass — это [метаязык](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D1%8F%D0%B7%D1%8B%D0%BA) на основе [CSS](https://ru.wikipedia.org/wiki/CSS), предназначенный для увеличения уровня абстракции CSS-кода и упрощения файлов [каскадных таблиц стилей](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D0%BA%D0%B0%D0%B4%D0%BD%D1%8B%D0%B5_%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%8B_%D1%81%D1%82%D0%B8%D0%BB%D0%B5%D0%B9).*) Все файлы стилей подключаются в файле app.scss*

***Слайд 16****В файле vars.scss находятся переменные, используемые в файлах стилей, а так же миксина (примесь, фрагмент кода используемый многократно) для медиазапросов (сокращает запись)*

***Открыть главную страницу сайта + Слайд 17****После того, как была закончена верстка, требовалось добавить эффекты анимации с помощью JavaScript. На главной странице и странице обо мне реализована анимация появления элементов контента в зависимости от положения страницы по оси Y с помощью свойства Window.pageYOffset (*возвращает количество пикселей, на которое прокручен документ по вертикали (вниз или вверх*). Горизонтальное навигационное меню всплывает при прокрутке страницы снизу вверх*

***Слайд 18****Так же на главной странице добавлено вращение карточек предложений при нажатии на кнопку подробнее.*

*Все используемые слайдеры реализованы с помощью готового решения owl carousel, установленного через менеджер пакетов (npm* который входит в состав Node.js.*)*

***Показать на сайте + Слайд19****Реализована система смены языка (англ и рус) При выборе языка нажимая на кнопку выбора js с помощью метода fetch оправляет на сервер сетевой запрос с информацией о выбраном языке, с помощью php устанавливаются файлы куки с выбраным языком. (куки требуются для запоминания выбранного языка при перезагрузке страницы)*

***Слайд20*** *В атрибуте элементов* *switchable-text , для которых необходима смена языка добавлен текст на русском языке. (по умолчанию сайт открывается на английском*

***Слайд21****при получении ответа от сервера выполняется функция* ***switchLang() слайд19,*** *в которой происходит смена текстов из innerHTML в атрибут switchable-text в зависимости от атрибута* switch-lang ***Слайд20,*** *который меняется получая данные с сервера выполнением функции switchLang()* ***слайд20***

*Во**избежание лагов всем изменяемым язык элементам добавлен стиль opacity 0, который с задержкой меняется на 1 в процессе выполнения сценария перестановки атрибутов*

***показать на странице появление и смену попапов****Система входа и регистрации. Реализована нажатием на кнопку вход – открывается всплыв меню логин в котором можно переключиться на меню регистрации.*

***Слайд22 + 23****Предусмотрена валидация форм. На 23 слайде видно что валидация реализована в том числе с помощью регулярных выражений.*

***Слайд24****При успешном вводе данных если ошибок нет (errors.length = 0) Данные собираются в объект* ***params*** *и отпправляются на сервер методом fetch,*

***Слайд25****далее на сервере обрабатывается функцией addUser() в классе User.php  
(***Класс** - это базовое понятие в объектно-ориентированном программировании ([ООП](http://www.php.su/learnphp/phpoo/?basic)). Классы образуют синтаксическую базу ООП. Экземпляр класса - это **объект**. Объект - это совокупность данных (свойств) и функций (методов) для их обработки.*)  
Происходит повторная проверка полученных данных на соответсвие рег выражениям и дублирования email, пароль шифруется методом sha256 и в случае отсутсвия ошибок****Слайд26****глобальной переменной Session присваиваются значения полученных имени и емэил, а так же с помощью sql запроса в БД user добавляются данные регистрации, введенные пользователем* ***слайд27***

***Слайд28****В случае успешного занесения данных в БД функция addUser() возвращает success = true. Происходит смена соответствующих элементов страницы без перезагрузки.* ***Показать на примере зарегистрироваться на сайте.*** *На email пользователя и админа отправляется email об успешной регистрации.*

***Слайд29*** *В случае ошибки в окне регистрации появляется ошибка (например пользователь с таким именем уже существует)*

*Вход на сайте осуществляется по тому же принципу что и регистрация.*

***Показать на сайте меню*** *После входа пользователя на сайт становится доступно меню пользователя, состоящее из пунктов мои занятия, настройки, и выход из системы (в дальнейшем предусмотрен пункт сообщения)*

***Показать окно настроек****Настройки пользователя было решено сделать не в отдельной странице а во всплывающем меню. Открыв настройки мы видим возможность смены имени, пароля, почты, скайпа и аватара. JS функция changeSettings()*

***слайд30*** *осуществляет валидацию форм, проверяются различные условия например проверка на пустоту – если поле ввода пустое (исключая пробелы методом trim()) то переменной чек, которая является элементом вспылвающей подсказки и предыдущим братом (prevsibiling) поля ввода присваивается значение checkText = 'поле не должно быть пустым'; в зависимовсти от выбранного языка.*

***Слайд31*** *а так же отправляет на сервер введенные пользователем данные с помощью библиотеки axios (*При использовании Fetch приходится иметь дело с двумя промисами. А вот при работе с Axios у нас есть прямой доступ к JSON-результату в свойстве data объекта ответа. Использование Axios позволяет избежать написания больших объёмов шаблонного кода и сделать код чище и понятнее.*)*

***слайд32*** *обрабатываются функцией changeSettings() в классе User.php. При смене пароля требуется сначала ввести старый пароль а затем двукратное подтверждение нового.*

***Слайд33*** *При смене почты на новый введеннгый емэил отправляется письмо с кодом подтверждения, генерируемым случайным образом $code = rand(1000, 9999); который необходимо ввести в поле. В случае успеха новая почта добавляется в БД user. Отправка почты осуществляется с помощью библиотеки Phpmailer, используя почтовый сервер gmail.*

***Слайд34  
https://foliotek.github.io/Croppie/*** *Смена аватара реализована с помощью готового решения croppie. При добавлении аватарки в БД users заносятся путь к файлу аватарки для дальнейшей вставки в других элементах сайта.*

***Открыть страницу курсы****Страница курсы пока не реализована из-за отсутствия контента (заказчик еще не выпустил курсы) есть только дизайн.*

***Индивидуальные занятия****Перейдем к самой сложной части сайта – календарь бронирования. На странице занятие с преподавателем мы видим информацию о предлагаемом продукте, время урока и цену (цена меняется по усмотрению админа из админ-панели). Данный компонент реализован с помощью фреймворка vue.js (***Vue полностью подходит и для создания сложных одностраничных приложений**, появился в 2014 году, проекты на vue: Alibaba, Baidu, Xiaomi, Sina Weibo и др. Он входит в ядро Laravel и PageKit. Недавно свободная система управления репозиториями GitLab тоже перешла на Vue.js., В конце сентября 2016-го вышел в релиз Vue.js 2.0, еще круче и с упором на производительность — теперь используется виртуальный DOM*)*

*В этом компоненте реализованы следующие функции:   
переключение недель, подсветка текущего дня недели  
автоматическое изменение стилей временных интервалов в зависимости от их состояния в соответствии с указанными в компоненте тегами (если дата интервала уже истекла он становится недоступным, )****показать на примере как бронируется урок*** *выбор свободного интервала времени (одного или нескольких) и собственно бронирование нажатием на кнопку забронировать. Если пользователь не указал скайп – появление окна для добавления скайпа. После успешной отправки скайпа в бд, пользователь отправляется на страницу оплаты payment.php, на которой он видит информацию о выбранных занятиях, общую сумму к оплате, реквизиты получателя, ссылку для быстрой оплаты paypal при нажатии на которую сразу открывается сайт paypal с необходимыми реквизитами для оплаты, а так же инструкцию по оплате нажав на знак вопроса.  
Если пользователь передумал оплачивать бронь (не нажал на кнопку подтвердить по той или иной причине) его бронь подсвечивается красным , что означает неоплаченное занятие, нажав на которое появлется окно выбора действий – отменить либо оплатить. В случае бездействия пользователя все неоплаченные интервалы удаляются через 15 минут.  
 После оплаты пользователь подтверждает оплату нажав на соответствующую кнопку и получает сообщение на экране с дальнейшими инструкциями и email на почту с инфо о бронировании. Ссылка для перехода на страницу мои уроки.  
Вернувшись к календарю пользователь видит что занятия получили желтый цвет , и теперь требуют проверки и подтверждения оплаты преподавателем , которому так же отправлено письмо с информацией о бронировании (время, дата, имя и контакты пользователя).*

***Слайд35*** *Что бы реализовать изменения времени в зависимости от нахождения пользователя в том или ином поясе, потребовалось создать еще одну таблицу БД temp-gmt. Функция getIntervalsFromDB() получает данные из БД time-intervals (данные в которую добавляются админом),*

***Слайд36****Затем эти данные принимает функция addToTempDB(), которая обрабатывает данные в зависимости от часового пояса (delta – вычисляется как разница между текущим часовым поясом и поясом, полученным из БД).*

***Слайд37****Она плюсуется к разбитой на части строкой временного интервала (к части часа) и затем снова собирается в требуемый вид (13:00-14:00) и добавляется в массив, который в свою очередь добавляется в объект obj 1, 2, 3 в зависимости от условия (a < 0, … )*

***Слайд38****Затем вcе объекты добавляются в общий массив и отправляются в БД temp-gmt.*

***Слайд39****Затем данные из временной БД получает функция getOfTempDB(), производится проверка – сравнивается количество базовых интервалов в БД time-intervals и количество полученных из временной БД, если совпадает –* ***слайд40*** *данные обрабатвыаются*

***Слайд4****1  
И вставляются в календарь*

***Слайд42*** *На стороне сервера данные обрабатываются классом Calendar.php*

*Если проверка не выполняется то функция getIntervalsFromDB снова выполняет сама себя (рекурсивно)*

***Слайд43****Далее требовалось отобразить забронированные интервалы по всем пользователям (сделать недоступными для броин) и по текущему пользователю. В рамках функции getOfTempDB() выполняется запрос на получение данных из БД bookstime. Эти данные обрабатываются таким же образом как и в предыдущем случае (в зависимости от часового пояса вычисляется delta) добаввляются в массив bookedGmtArray* ***слайд44***

***Слайд45****После чего массив перебирается и сопоставляется с интервалами в календаре, и в зависимости от условий к интервалам применяются стили (оплачено не оплачено не подтверждено недоступно)*

***Слайд46****Так же предусмотрено автоматическое удаление неподтвержденных бронирований через 15 минут (если пользователь нажал забронировать и не оплатил урок) при бронировании в раздел таблицы bookingTime БД bookstime добавляется время на момент бронирования , в случае если delta становится больше 15 – бронь удаляется из БД.*

***Speaking-club****На странице спеакинг клаб такой же функционал как и в предыдущей, тольк*

***Открыть страницу мои уроки*** *Так же реализована страница мои уроки , доступная из меню пользователя, здесь указаны предстоящие уроки*

***Открыть главную страницу****.  
Каждая страница сайта (кроме админ-панели) состоит из шапки и футера.В шапке находится соцсети, логотип, кнопки смены языка и входа на сайт и горизонтального навигационного меню, которое всплывает при прокрутке страницы снизу вверх. (на мобильной версии шапка всегда остается зафикисрованной, а нав меню трансформируется в бургер меню).   
Футер сайта так же имеет логотип, емэил для связи с преподавателем и декорирован тематическими иконками елочек)*

1. **Тестирование**

[*https://habr.com/ru/company/vdsina/blog/522932/*](https://habr.com/ru/company/vdsina/blog/522932/)

1. **Планы по доработке**

***Месенджер, эквайринг, сео оптимизация, почтовая рассылка, компонент отзывов, рефакторинг кода – улучшение качества и уменьшение количества.***

***Рефа́кторинг****(англ.****refactoring****), или перепроектирование****кода****, переработка****кода****, равносильное преобразование алгоритмов — процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы.*

*Рефакторинг – из-за неопытности в коде присутствует множество повторений, некоторые функции продублированы повторно в разных компонентах.*

*Месенджер -*

1. **Заключение**

***Срок работы над сайтом составил 3 месяца***

***Чему научился?***

***Что понравилось?***

***Чем хочу заняться дальше?***

*В дальнейшем планирую углубляться в ООП и компон.ориент.прогр.КОП*

*ООП -*  [методология программирования](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), основанная на представлении программы в виде совокупности [объектов](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), каждый из которых является экземпляром определённого [класса](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), а классы образуют иерархию наследования

*КОП -* опирающаяся на понятие **компонента** — независимого модуля [исходного кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) программы, предназначенного для [повторного использования](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0) и развёртывания и реализующегося в виде множества языковых конструкций (например, «[классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))» в [объектно-ориентированных](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [языках программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), объединённых по общему признаку и организованных в соответствии с определёнными правилами и ограничениями.

**Сайт полностью рукописный, из готовых решений использовался только слайдер owl carousel**