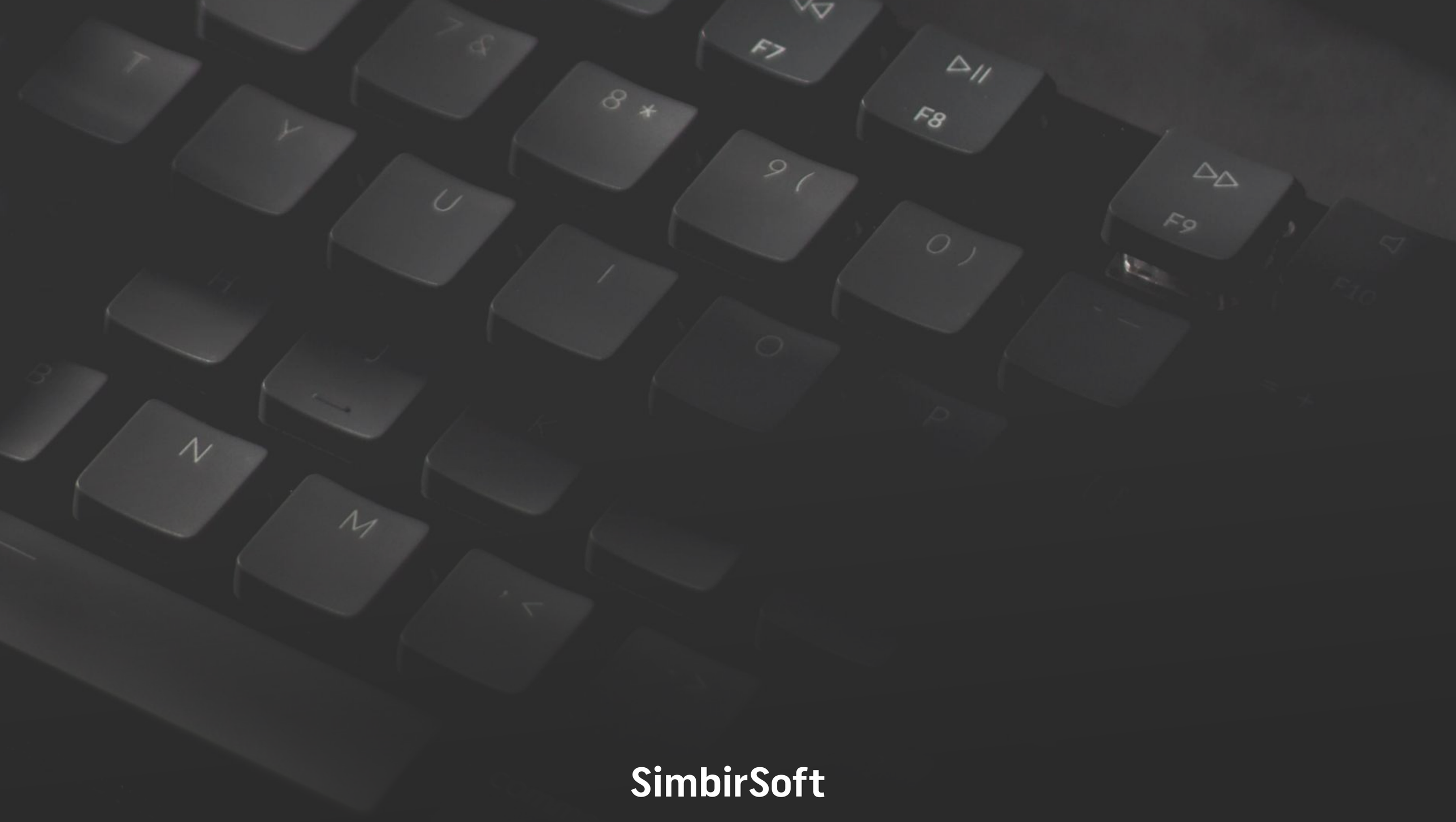

SimbirSoft

Рефакторинг личного кабинета без остановки релизов

Как ускорить разработку в 4 раза, снизить на 90% критические ошибки и сэкономить 200+ часов трудозатрат за первый месяц

Константин, руководитель отдела фронтенд-разработки с 10-летним опытом работы. За последние 2 года команда под его руководством провела 7 успешных рефакторингов* в корпоративных проектах без остановки релизов. Суммарная экономия бюджета заказчиков — 40 млн ₽

«Рефакторинг страшно начинать — кажется, что остановится вся разработка. На деле можно обновлять интерфейс и выпускать релизы без простоев. Представляем вам инструкцию для запуска цифровых продуктов на основе реализованного кейса «Умная крыша» для компании «ТЕХНОНИКОЛЬ»

Пользуясь рекомендациями из 7 шагов, вы сможете:

- оценить реальную экономию времени и затрат на адаптацию сотрудников при запуске проекта
- спланировать сроки вывода MVP — минимально жизнеспособного продукта — и повлиять на скорость выпуска проекта
- проверить теорию гипотез в ходе реализации проекта

*Рефакторинг — оптимизация кода без изменения функциональности

Шаг 1

Аудит «как есть» — без этого не начинайте

Что сделать:

Составить карту всех модулей личного кабинета (информационные разделы, баланс, финансовые данные, заказы, управления подписками, хранение данных, отчёты и т.д.)

Выявить самые «тяжёлые» с точки зрения кода и ошибок страницы.

Определить, где технический долг бьёт по скорости релизов больше всего.

Результат: список приоритетов — с чего начинать рефакторинг, чтобы сразу увидеть эффект.

Шаг 2

Выберите «безопасную зону» для первого захода

Что сделать:

Найти модуль, который меняется редко, но тормозит систему.

Сделать рефакторинг именно там — как обкатку процесса.

Не трогать критический функционал (например, финансовые данные, баланс) в первой итерации.

Результат: быстрая победа без риска — команда нарабатывает уверенность.

Шаг 3

Разделите рефакторинг и новую функциональность

Золотое правило:

НЕ делайте рефакторинг и добавление новых функций в одной задаче.

Задача = или улучшаем старый код, или пишем новое. Не вместе.

Как организовать:

- Часть команды (или часть спринта) — только на рефакторинг.
- Вторая часть команды — на новые релизы.

Результат: вы не тормозите выход новых фич на 2 недели.

Шаг 4

Внедрите feature toggle «переключатель функций» — ваш главный инструмент безопасности

Что это:

Новый код пишется рядом со старым, но выключен для пользователя.

Включаете его только когда все протестировано и готово.

Пример. Как мы делали на «Умной крыше»:

- Новую карту намоканий разрабатывали параллельно со старой.

- Переключили пользователей только после 2 недель стабильной работы.

Результат: пользователи не видят «сырой» интерфейс, а вы не боитесь сломать рабочую среду.

Шаг 5

Автотесты — не роскошь, а страховка рефакторинга

Что сделать минимум:

- Написать smoke-тесты (т.е. выполнять быструю проверку основных функций ПО) для критического пути: вход → просмотр датчиков → открытие карты.
- Перед каждым релизом прогонять их автоматически.
- Пример. Как это было на проекте «Умная крыша»: на старте у нас тестов не было. Мы построили процессы тестирования с нуля — и это снизило критические ошибки на 90%.
- Результат: вы смело меняете код, потому что тесты скажут, если что-то сломалось.

Шаг 6

Рефакторинг частями, а не «большим взрывом»

Что сделать минимум:

Плохой подход: «Давайте перепишем весь личный кабинет за месяц».
Хороший подход: «В этом спринте рефакторим страницу датчиков, в следующем — шапку и навигацию».

Как мы делали:

- Редизайн и рефакторинг проводили модуль за модулем.
- После каждого — сразу релиз и сбор обратной связи.

Результат: вы не накапливаете риск «а вдруг всё сломается при финальной сборке».

Шаг 7

Ритм релизов — ваш лучший индикатор безопасности

Что делать:

Выпускайте релиз каждую неделю, даже если изменений мало.

Если релиз прошёл гладко — вы контролируете ситуацию.

Если начались проблемы — вы узнаете об этом максимум через неделю, а не через месяц.

Пример. На «Умной крыше» мы вышли на 1 релиз в неделю.
Это и есть доказательство, что рефакторинг не оказал негативного влияния на скорость обновлений.

Типичные ошибки при самостоятельном рефакторинге.

Почему большинство команд проваливают рефакторинг? Они пропускают шаг № 1 или № 5 из-за нехватки времени, а потом вынуждены исправлять ошибки. Готовая команда подключается уже с выстроенными процессами тестирования и готовым набором инструментов для работы.

Что дальше?

Рекомендуем не строить команду с нуля и нанимать разработчиков по отдельности. Обратитесь к нашей готовой команде! Это не фрилансеры из биржи. Это наши штатные Frontend-, Backend-разработчики, QA-инженеры, бизнес-аналитики и другие опытные специалисты под ваш запрос и технологический стек.

Вы получаете:

- Старт разработки в первые дни вместо 4–8 недель на поиск
- Выход на продуктивность за дни, а не недели
- Предсказуемую скорость и качество
- Снижение нагрузки на менеджмент
- Бесшовную замену без остановки работ
- Юридическую безопасность — риски занятости на нас

Если описанные проблемы вам знакомы, не приступайте к рефакторингу без предварительного аудита и подготовки.

За 2 часа проведём экспресс-аудит личного кабинета или раздела сайта: оценим удобство интерфейса, выявим проблемные зоны и подготовим отчёт с рекомендациями по улучшению.

Переходите по ссылке и оставляйте заявку на экспресс-аудит!