HolyJS 2026 Autumn

Schema Driven UI

в enterprise-разработке

Маргарита Козырева
Ведущий разработчик

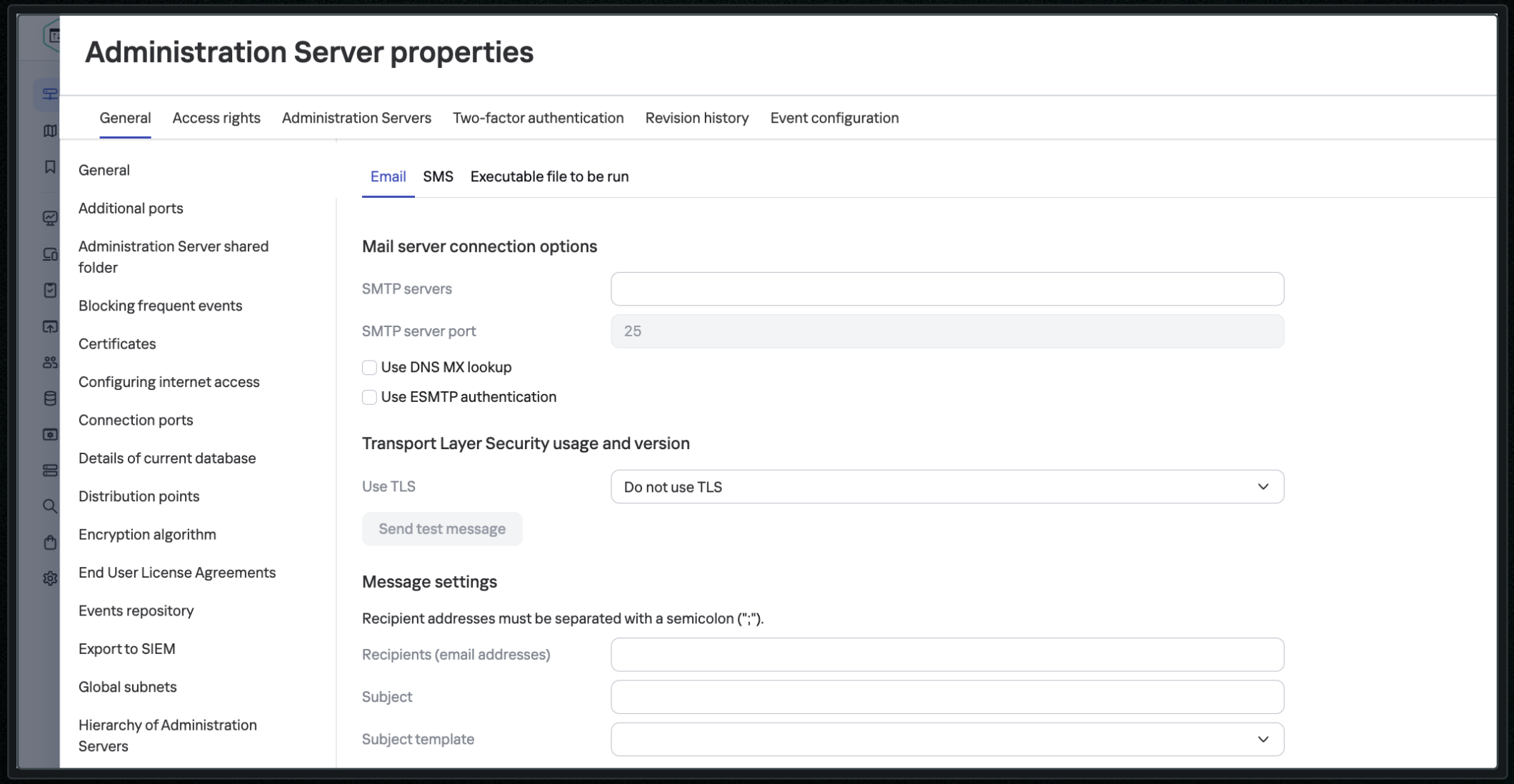
kaspersky





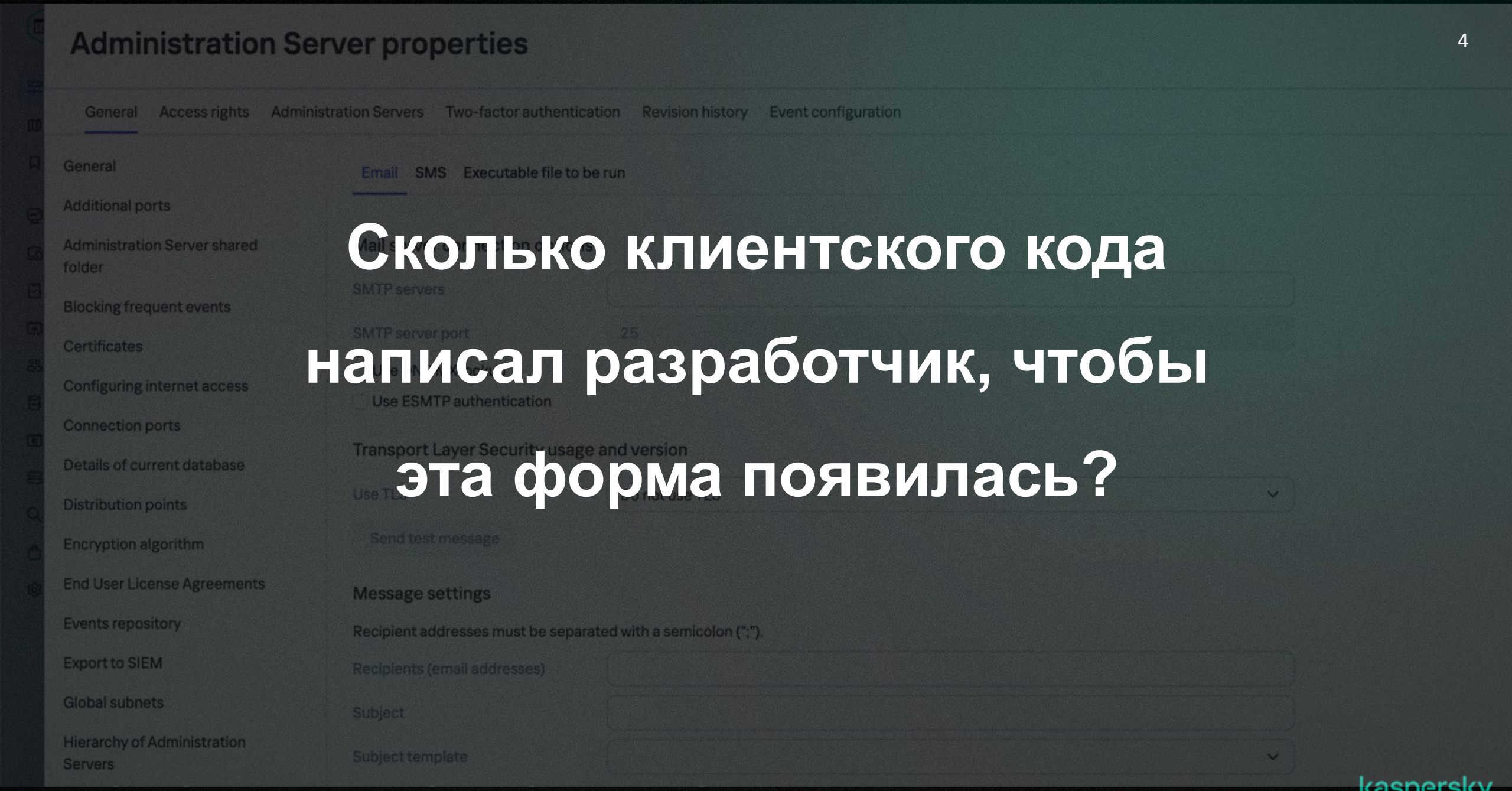
Маргарита Козырева

- Senior Frontend в Kaspersky
- 7+ лет во фронте
- Строила дизайн-системы с нуля
- Работала в VK, Positive Tech, X5, Raif
- Ушла из тимлидов в сеньоры

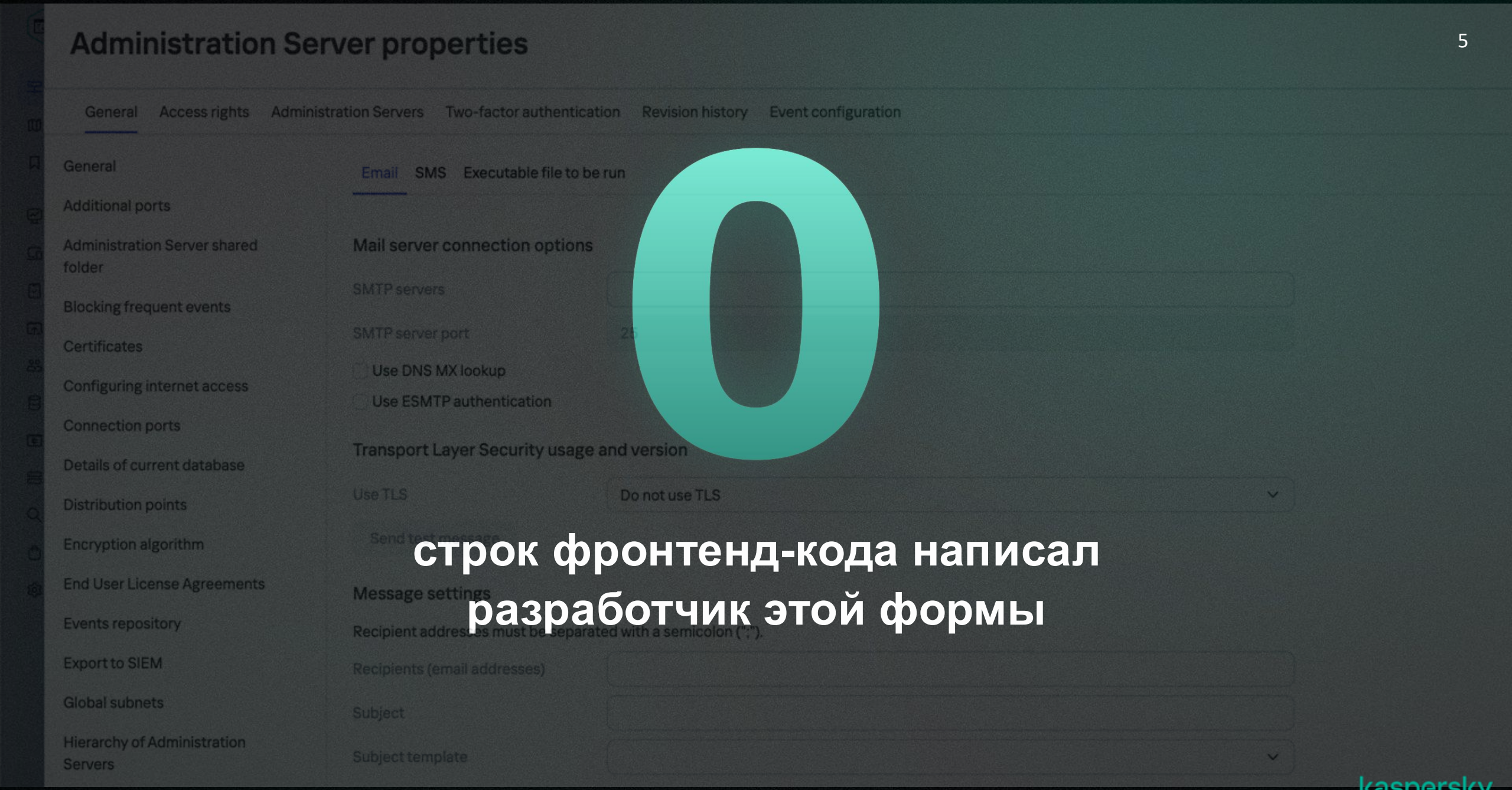


Настройки email-уведомлений: семь секций, два десятка полей, зависимости, валидация

kaspersky



Сколько клиентского кода написал разработчик, чтобы эта форма появилась?



0

строк фронтенд-кода написал разработчик этой формы

О чём поговорим

1 История возникновения



О чём поговорим

1 История возникновения

2 Про наш подход Schema Driven UI



О чём поговорим

1 История возникновения

2 Про наш подход Schema Driven UI

3 Преимущества



О чём поговорим

1. История возникновения
2. Про наш подход Schema Driven UI
3. Преимущества
4. Как это устроено: DSL, движок, миграции



О чём поговорим

1. История возникновения
2. Про наш подход Schema Driven UI
3. Преимущества
4. Как это устроено: DSL, движок, миграции
5. Причём тут снова AI?



О чём поговорим

1 История возникновения
2 Про наш подход Schema Driven UI
3 Преимущества
4 Как это устроено: DSL, движок, миграции
5 Причём тут снова AI?
6 Чего нам эта идея стоила и с чего начать, если захотите так же



Часть 1

История возникновения

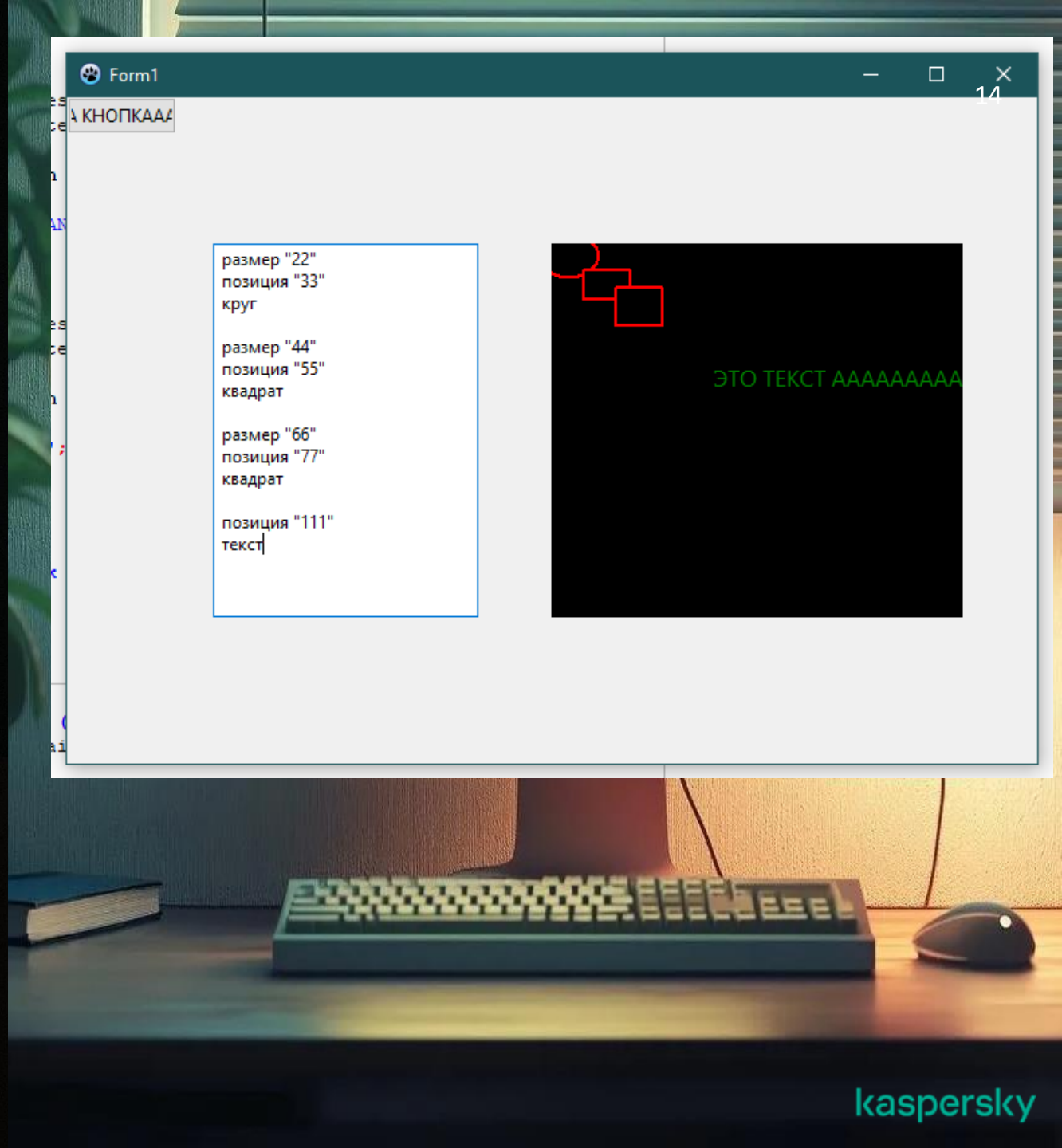


Идея «интерфейс не кодом, а описанием» старше многих из нас.

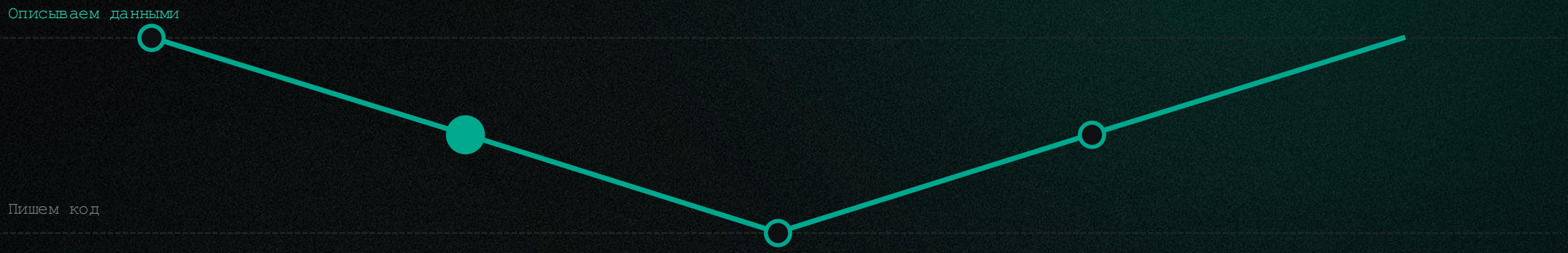
90-е

Delphi и Visual Basic. Кнопочки расставляли мышкой, форма сохранялась в текстовый ресурсный файл.

Основатель подхода



Один и тот же маятник, раз в десять лет



90-е

Delphi, VB

2000-е

PHP, ASP

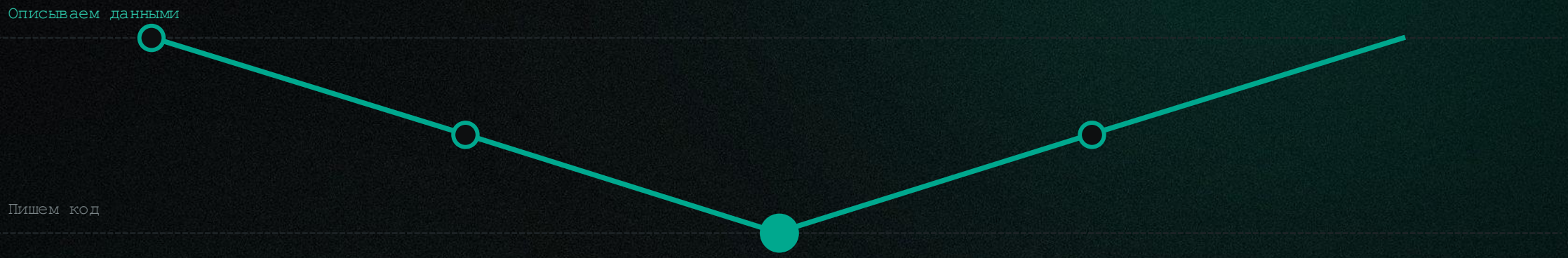
2000-е

Переход

Сервер сам рисует HTML - PHP, ASP



Один и тот же маятник, раз в десять лет



90-е

Delphi, VB

2000-е

PHP, ASP

2010-е

SPA, React

2010-е

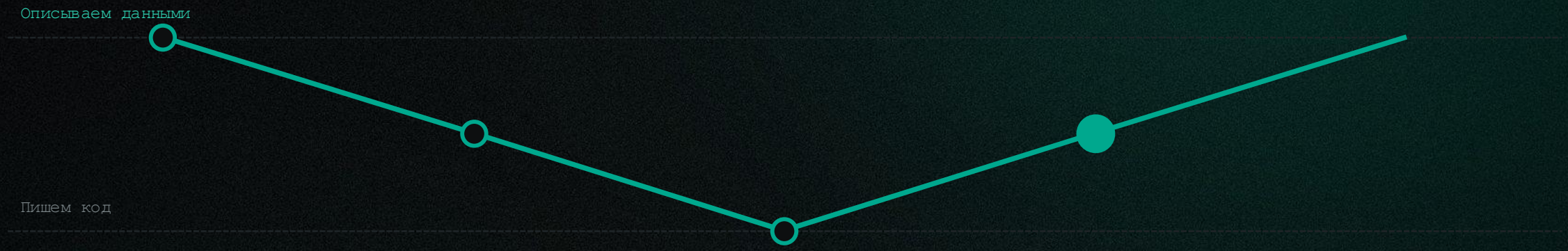
Код

SPA-революция забрала интерфейс на клиент.

КАЖДЫЙ! ЭКРАН! РУКАМИ!!!



Один и тот же маятник, раз в десять лет



90-е	2000-е	2010-е	2015+
Delphi, VB	PHP, ASP	SPA, React	JSON Schema

2015+

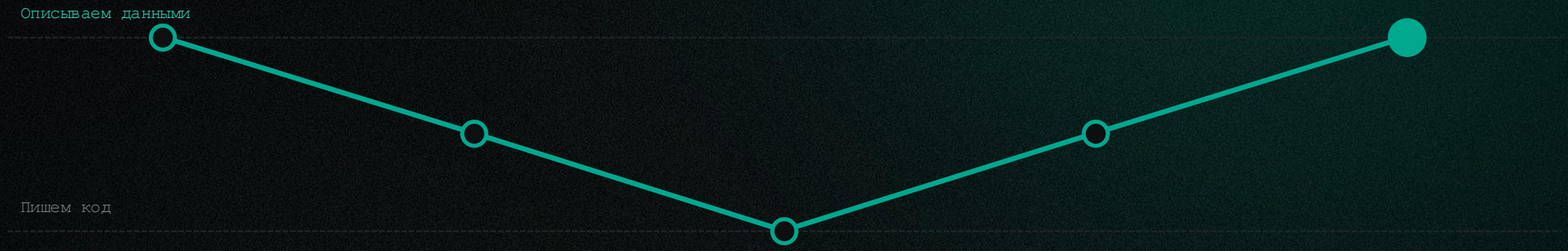
Маятник пошёл обратно

Генераторы форм по JSON Schema, конструкторы, low-code.

СТАЛО РЕАЛЬНО ДОРОГО!



Один и тот же маятник, раз в десять лет



90-е	2000-е	2010-е	2015+	2020-е
Delphi, VB	PHP, ASP	SPA, React	JSON Schema	BDUI

2020-е

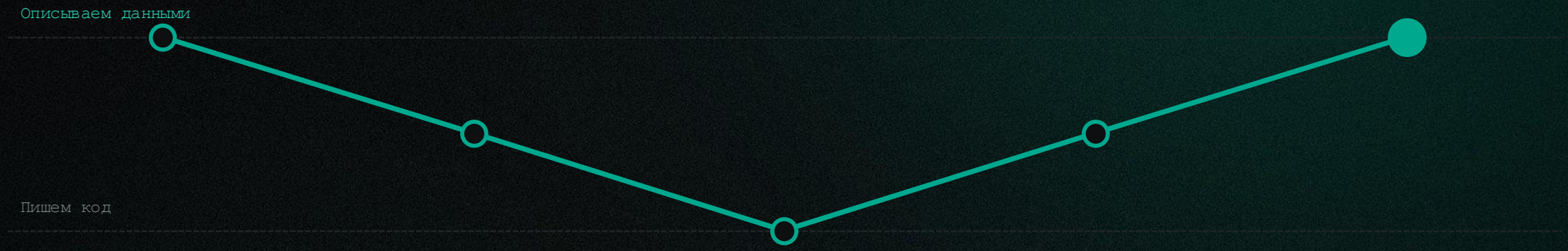
Backend Driven UI

Индустрия заново изобрела ресурсный файл из Delphi - только теперь это JSON, а движок живёт в браузере.

Сейчас



Один и тот же маятник, раз в десять лет

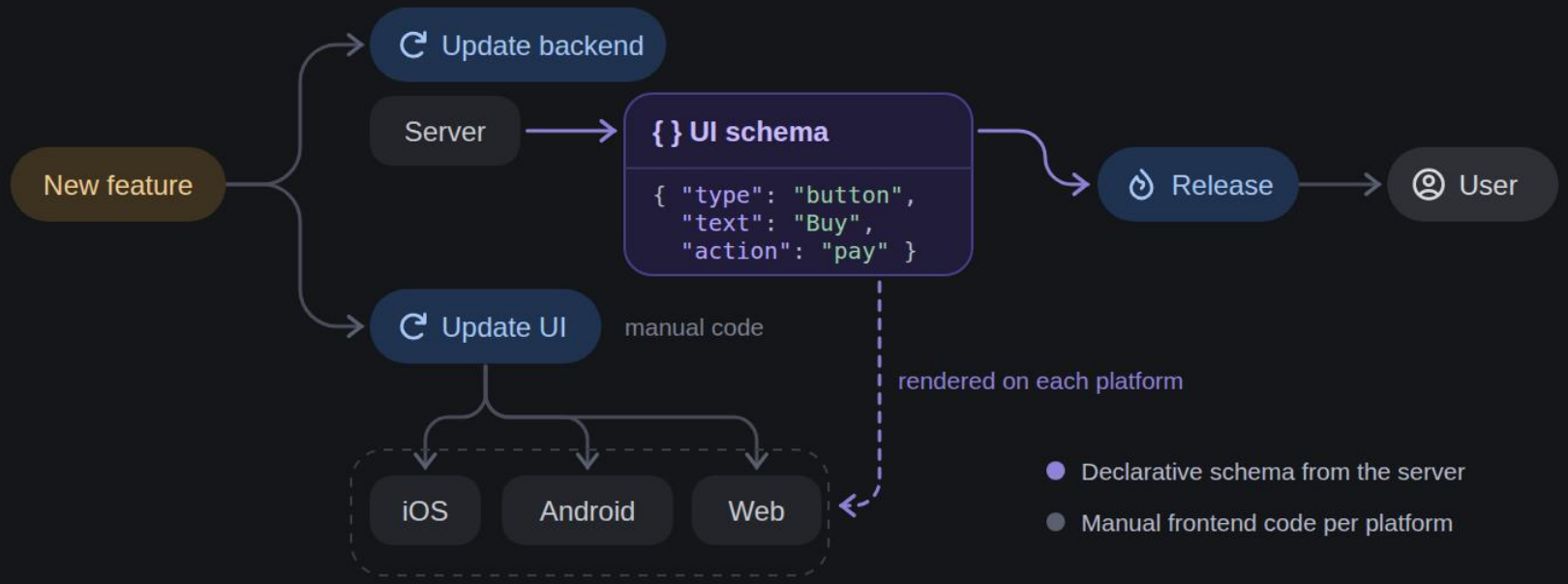


90-е	2000-е	2010-е	2015+	2020-е
Delphi, VB	PHP, ASP	SPA, React	JSON Schema	BDUI

Не наша выдумка - очередной ход маятника. Мы просто оказались на его пути раньше многих.

Backend Driven UI

- это когда интерфейс описывается декларативной схемой



Backend Driven UI
- это когда интерфейс описывается декларативной схемой

Схема отвечает на пять вопросов:

1

Какие элементы показать

Backend Driven UI
- это когда интерфейс описывается декларативной схемой

Схема отвечает на пять вопросов:

1

Какие элементы показать

2

Как они связаны с данными

Backend Driven UI
- это когда интерфейс описывается декларативной схемой

Схема отвечает на пять вопросов:

1

Какие элементы показать

2

Как они связаны с данными

3

Какие правила проверки

Backend Driven UI
- это когда интерфейс описывается декларативной схемой

Схема отвечает на пять вопросов:

1. Какие элементы показать
2. Как они связаны с данными
3. Какие правила проверки
4. Какие зависимости между полями

Backend Driven UI
- это когда интерфейс описывается декларативной схемой

Схема отвечает на пять вопросов:

1. Какие элементы показать
2. Как они связаны с данными
3. Какие правила проверки
4. Какие зависимости между полями
5. Какие действия доступны

Backend Driven UI
- это когда интерфейс описывается декларативной схемой

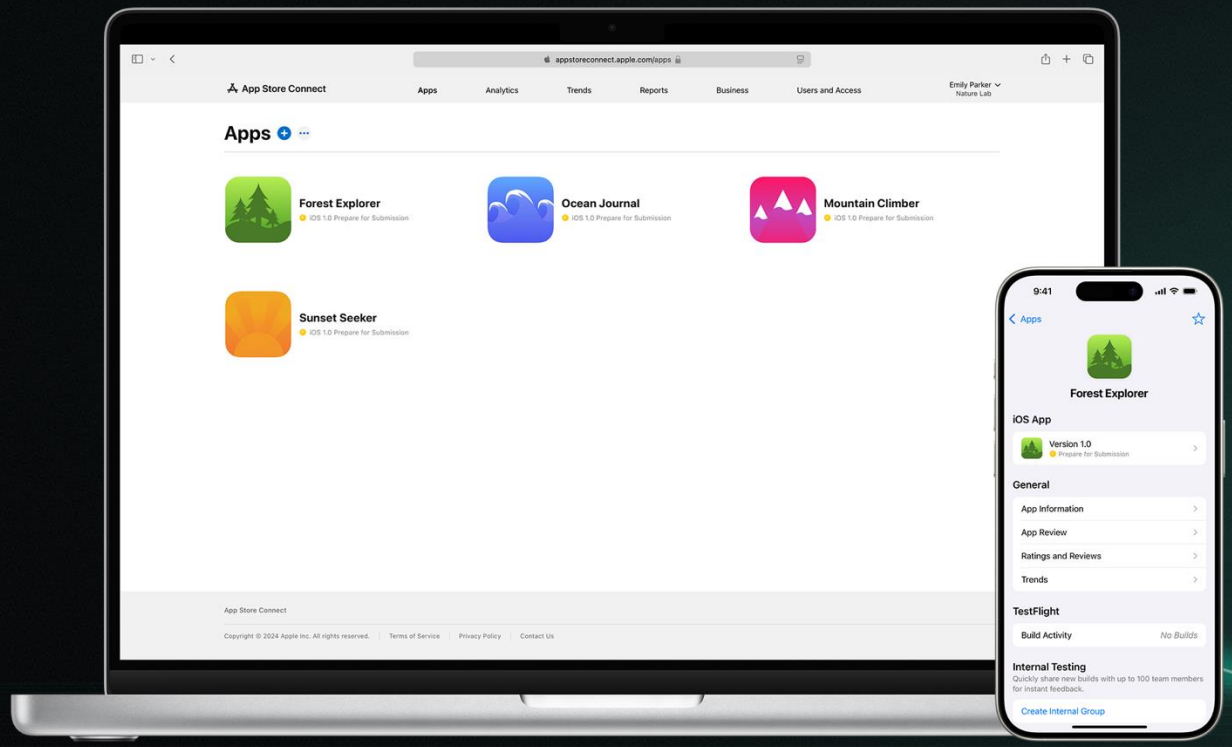
Схема отвечает на пять вопросов:

1	2	3	4	5
Какие элементы показать	Как они связаны с данными	Какие правила проверки	Какие зависимости между полями	Какие действия доступны

Поменял JSON на сервере - у всех изменился экран. Без релиза клиента.

Хайп пришёл из мобилок

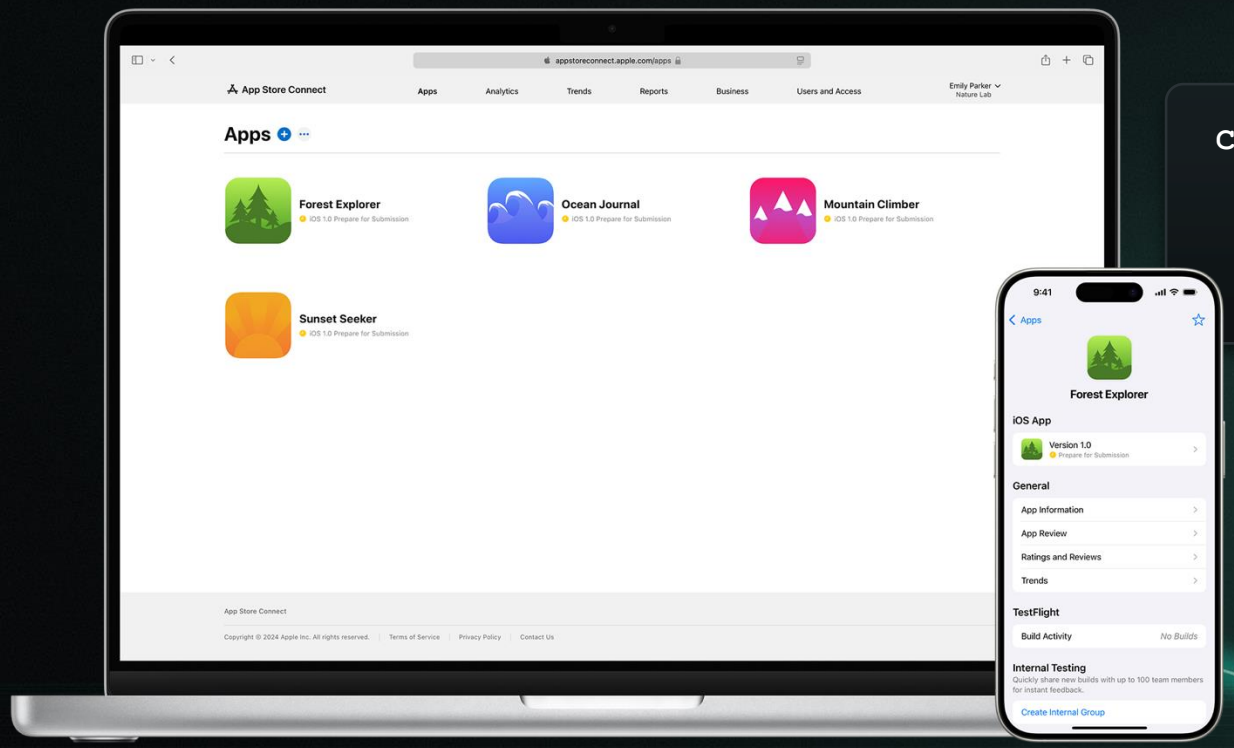
Причина была предельно практическая - ревью в сторах.



Waiting for Review

Хайп пришёл из мобилок

Причина была предельно практическая - ревью в сторах.



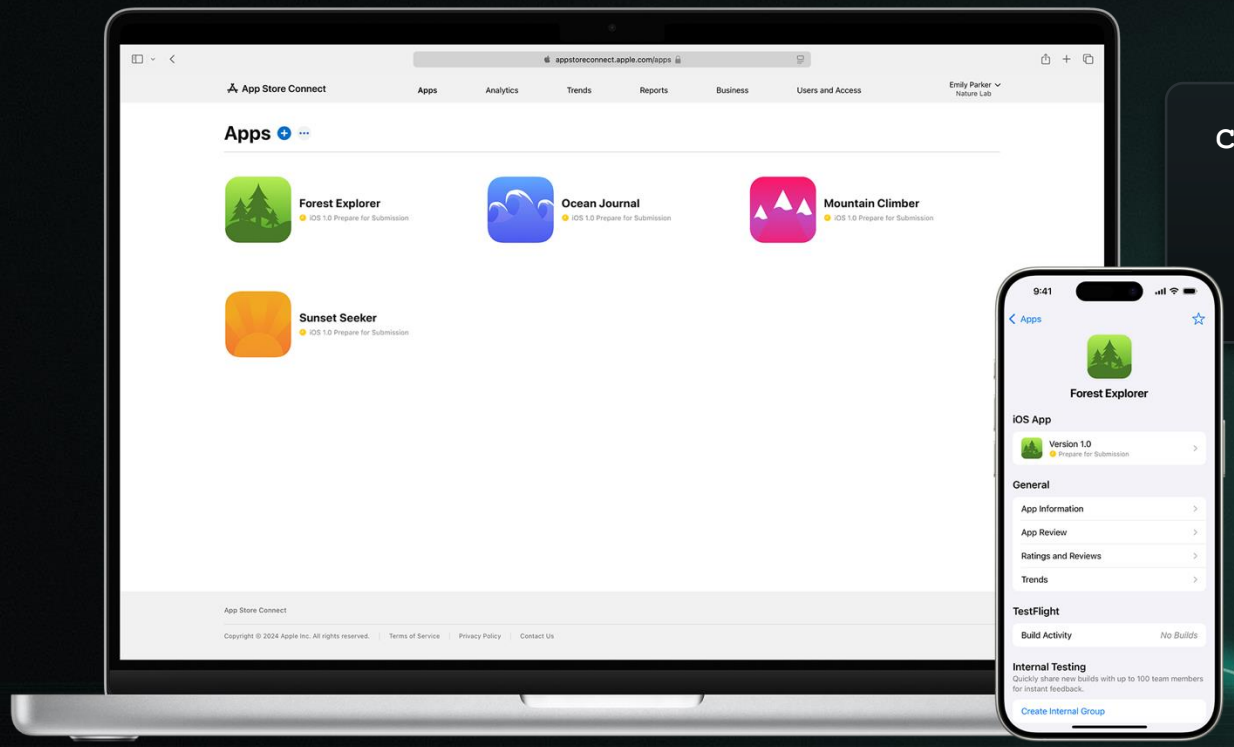
Сборка

Waiting for Review

kaspersky

Хайп пришёл из мобилок

Причина была предельно практическая - ревью в сторах.



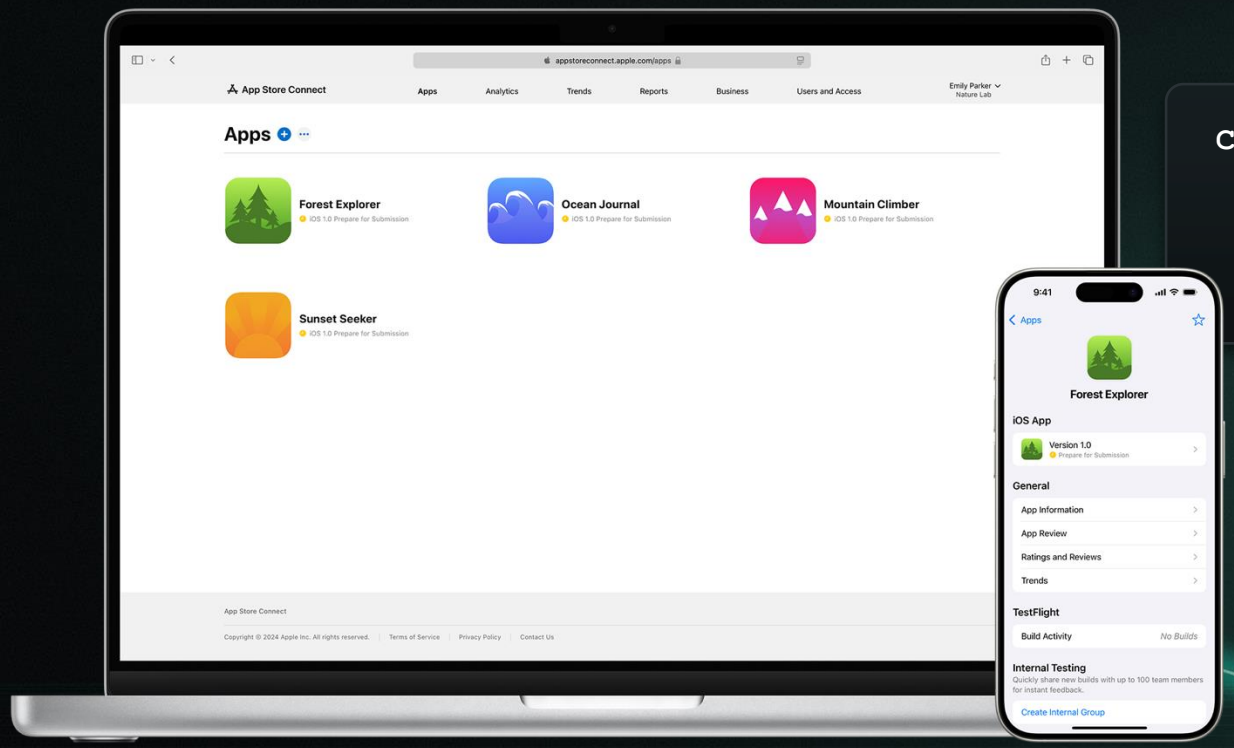
Сборка -> Релиз

Waiting for Review

kaspersky

Хайп пришёл из мобилок

Причина была предельно практическая - ревью в сторах.



Сборка -> Релиз -> Ревью

App Store

Waiting for Review

kaspersky

2–7

дней ревью на каждое изменение экрана

Именно эта арифметика и породила Backend Driven UI.

Сервер присылает не данные, а сам экран

Сервер

присылает описание экрана

kaspersky

Сервер присылает не данные, а сам экран

Сервер

присылает описание экрана

–>

Приложение

рисует нативными компонентами

kaspersky

Сервер присылает не данные, а сам экран

Сервер

присылает описание экрана

->

Приложение

рисует нативными компонентами

->

Пользователь

новый экран без релиза

kaspersky

Сервер присылает не данные, а сам экран

Сервер

присылает описание экрана

->

Приложение

рисует нативными компонентами

->

Пользователь

новый экран без релиза











Сервер присылает не данные, а сам экран

Airbnb

Lyft

Яндекс, DivKit

Авито

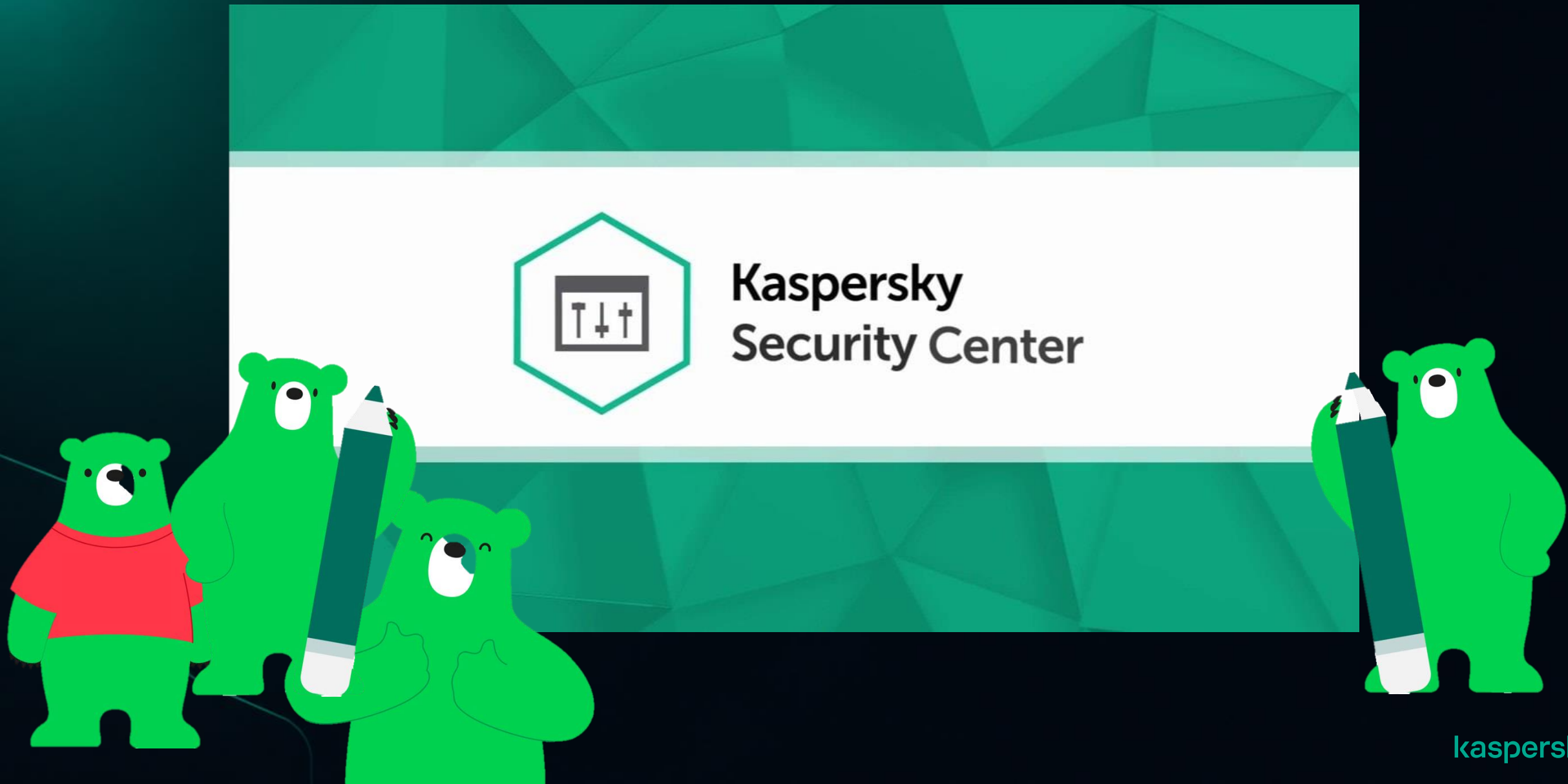
Драйвером была не красота архитектуры, а скорость изменений.

Часть 2

Сотни форм, десятки команд
Schema Driven UI

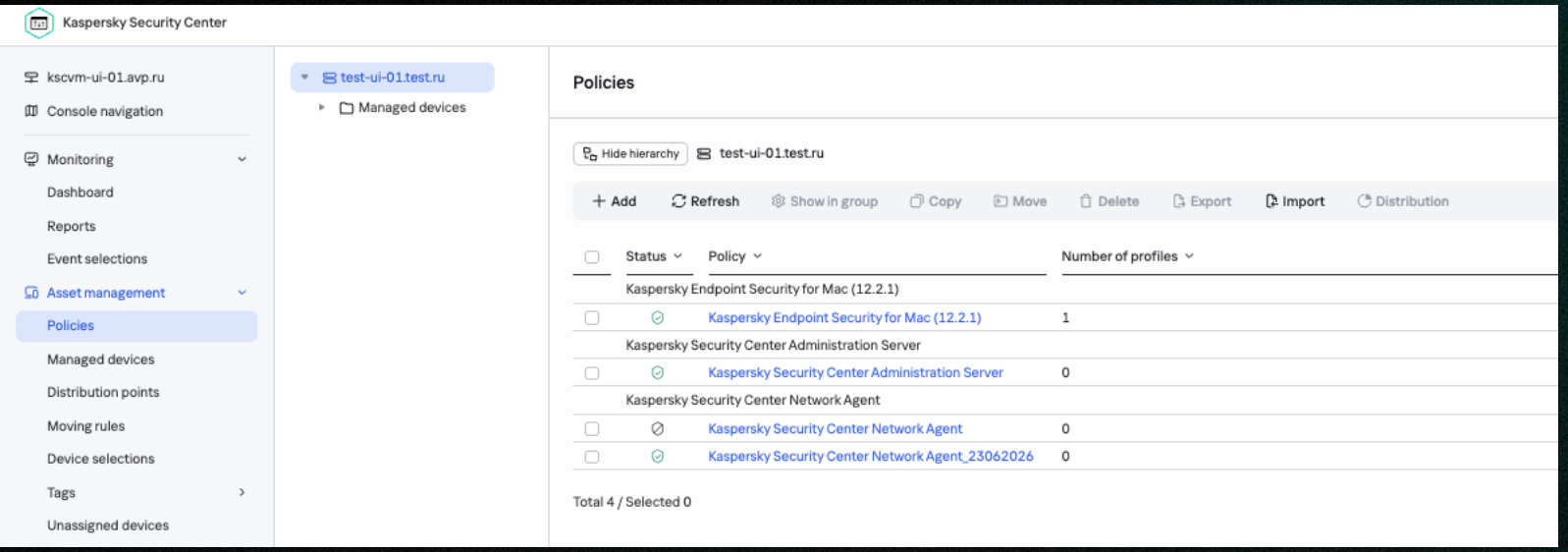


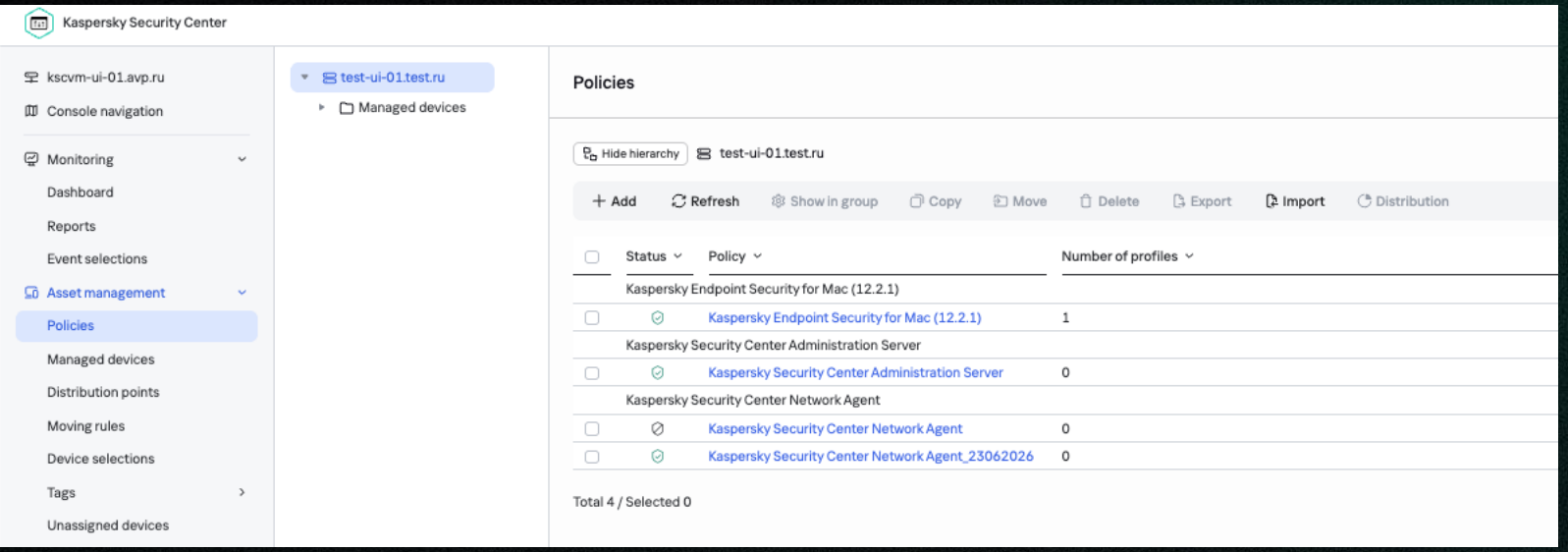
система для управления корпоративной защитой

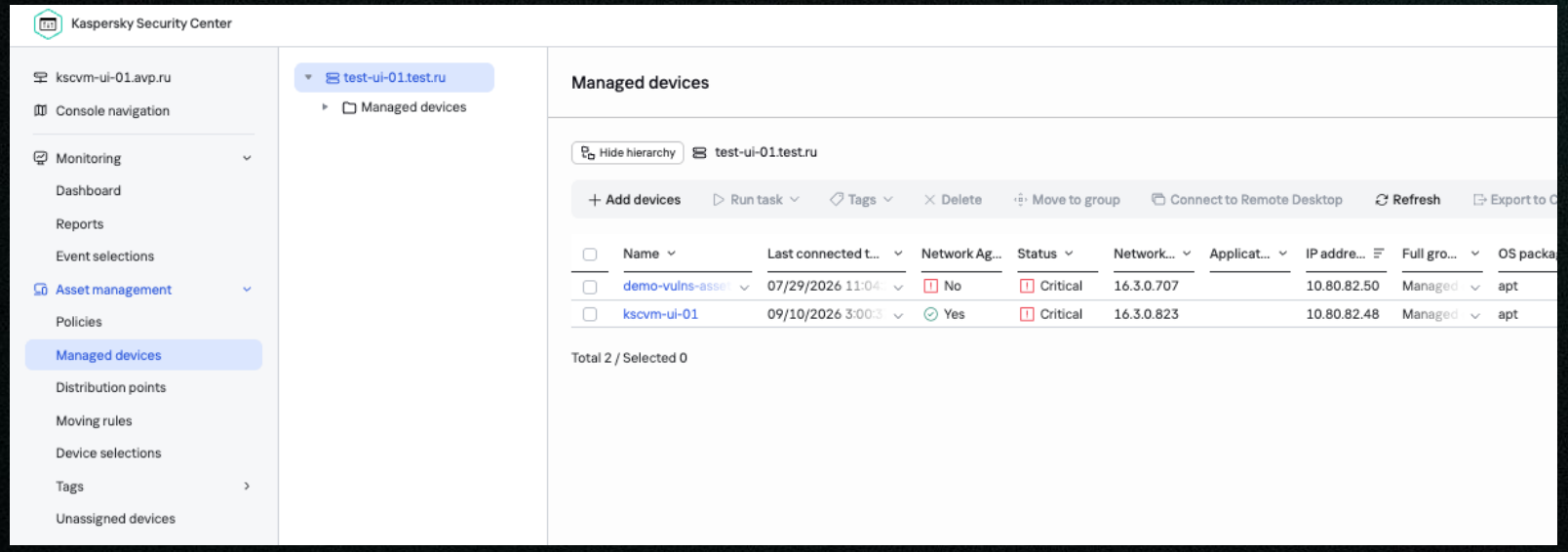


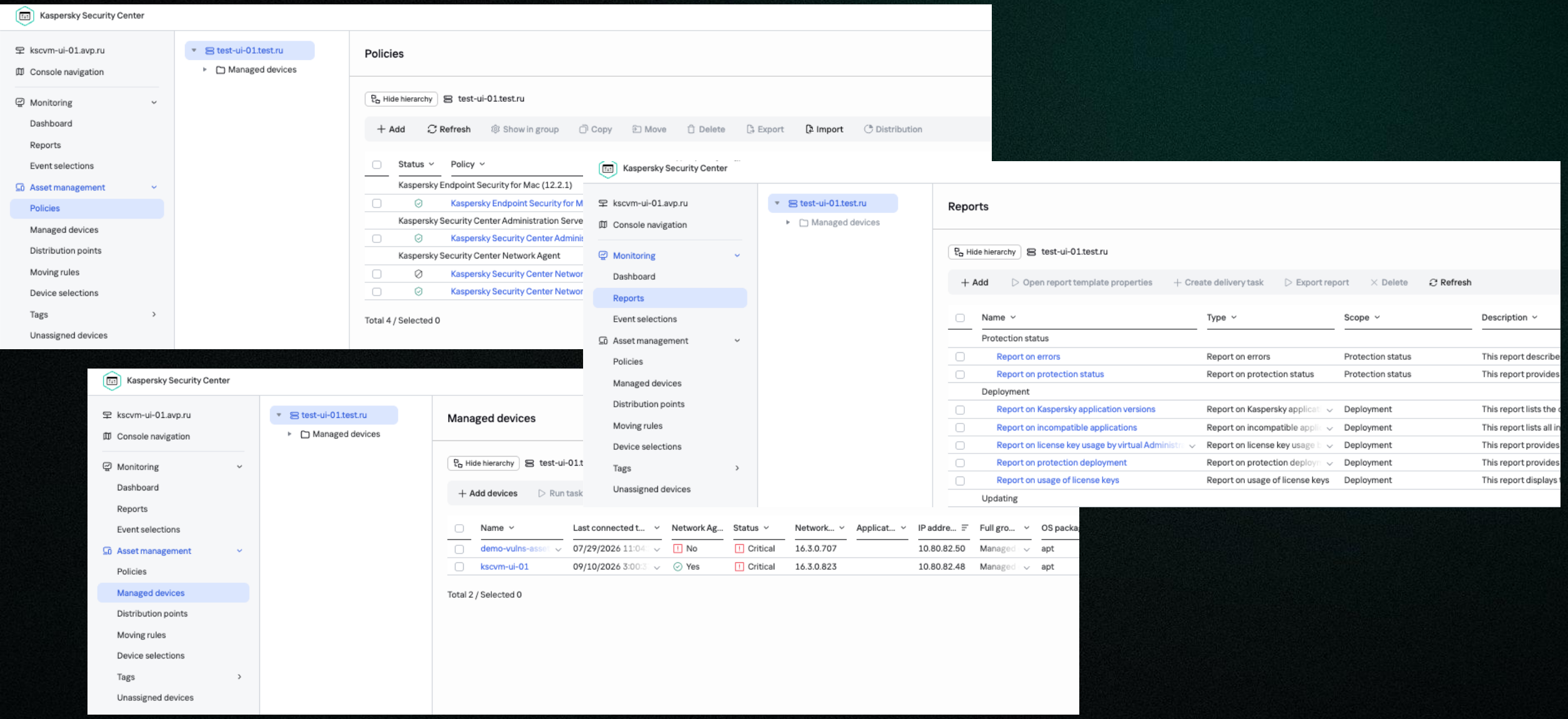
Сотни однотипных форм.
Десятки команд.
Один продукт, который должен выглядеть единым целым



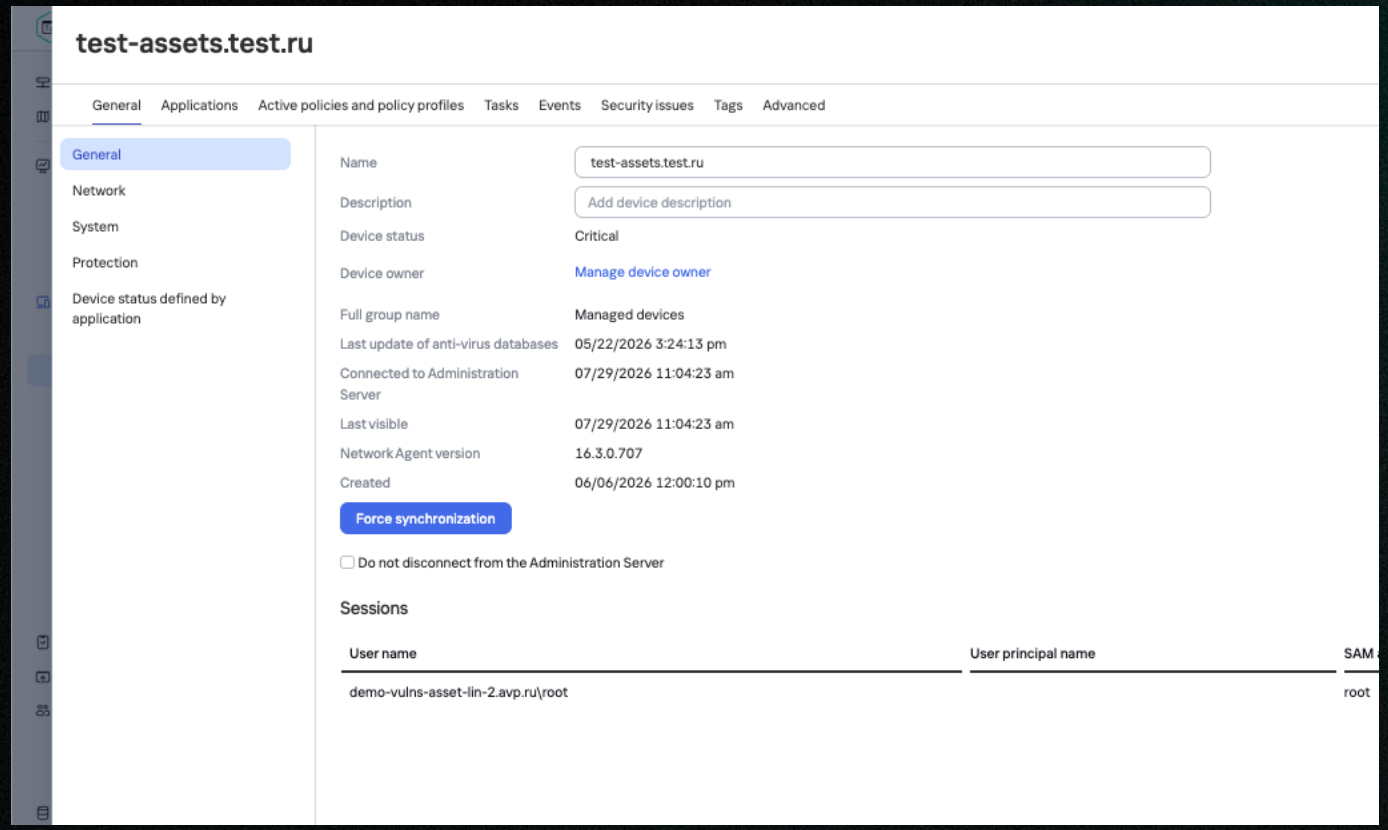

Kaspersky Security Center
kscvm-ui-01.avp.ru
Console navigation
Monitoring
Dashboard
Reports
Event selections
Asset management
Policies
Managed devices
Distribution points
Moving rules
Device selections
Tags
Unassigned devices
test-ui-01.test.ru
Managed devices
Policies
Hide hierarchy
test-ui-01.test.ru
Add
Refresh
Show in group
Copy
Move
Delete
Export
Import
Distribution
Status
Policy
Number of profiles
Kaspersky Endpoint Security for Mac (12.2.1)
Kaspersky Endpoint Security for Mac (12.2.1) 1
Kaspersky Security Center Administration Server
Kaspersky Security Center Administration Server 0
Kaspersky Security Center Network Agent
Kaspersky Security Center Network Agent 0
Kaspersky Security Center Network Agent_23062026 0
Total 4 / Selected 0

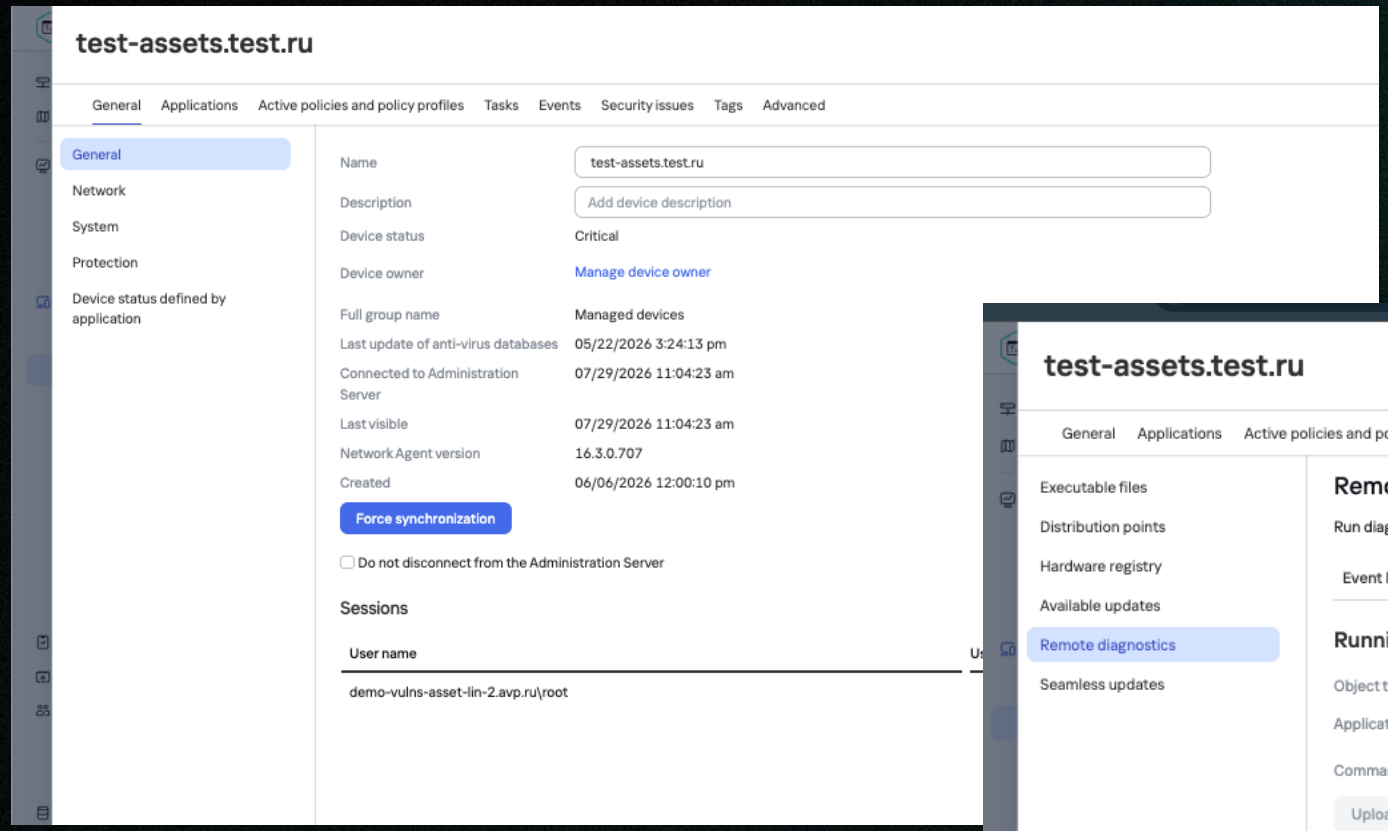

Kaspersky Security Center
kscvm-ui-01.avp.ru
Console navigation
Monitoring
Dashboard
Reports
Event selections
Asset management
Policies
Managed devices
Distribution points
Moving rules
Device selections
Tags
Unassigned devices
test-ui-01.test.ru
Managed devices
Policies
Hide hierarchy
test-ui-01.test.ru
Add
Refresh
Show in group
Copy
Move
Delete
Export
Import
Distribution
Status
Policy
Number of profiles
Kaspersky Endpoint Security for Mac (12.2.1)
Kaspersky Endpoint Security for Mac (12.2.1)
1
Kaspersky Security Center Administration Server
Kaspersky Security Center Administration Server
0
Kaspersky Security Center Network Agent
Kaspersky Security Center Network Agent
0
Kaspersky Security Center Network Agent_23062026
0
Total 4 / Selected 0

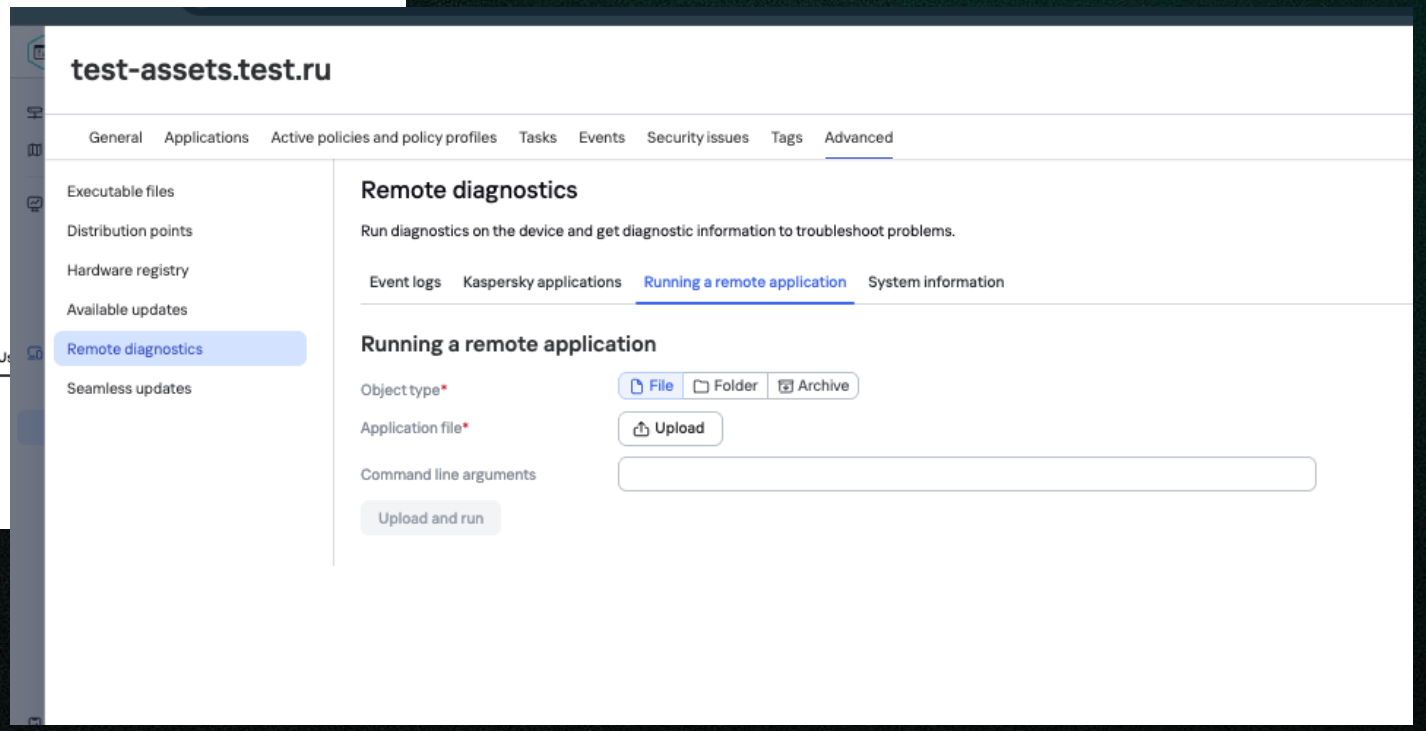

Kaspersky Security Center
kscvm-ui-01.avp.ru
Console navigation
Monitoring
Dashboard
Reports
Event selections
Asset management
Policies
Managed devices
Distribution points
Moving rules
Device selections
Tags
Unassigned devices
test-ui-01.test.ru
Managed devices
Managed devices
Hide hierarchy
test-ui-01.test.ru
Add devices
Run task
Tags
Delete
Move to group
Connect to Remote Desktop
Refresh
Name
Last connected t...
Network Ag...
Status
Network...
Applicat...
IP addre...
Full gro...
OS packa
demo-vulns-asse
07/29/2026 11:04
No
Critical
16.3.0.707
10.80.82.50
Managed
apt
kscvm-ui-01
09/10/2026 3:00:3
Yes
Critical
16.3.0.823
10.80.82.48
Managed
apt
Total 2 / Selected 0

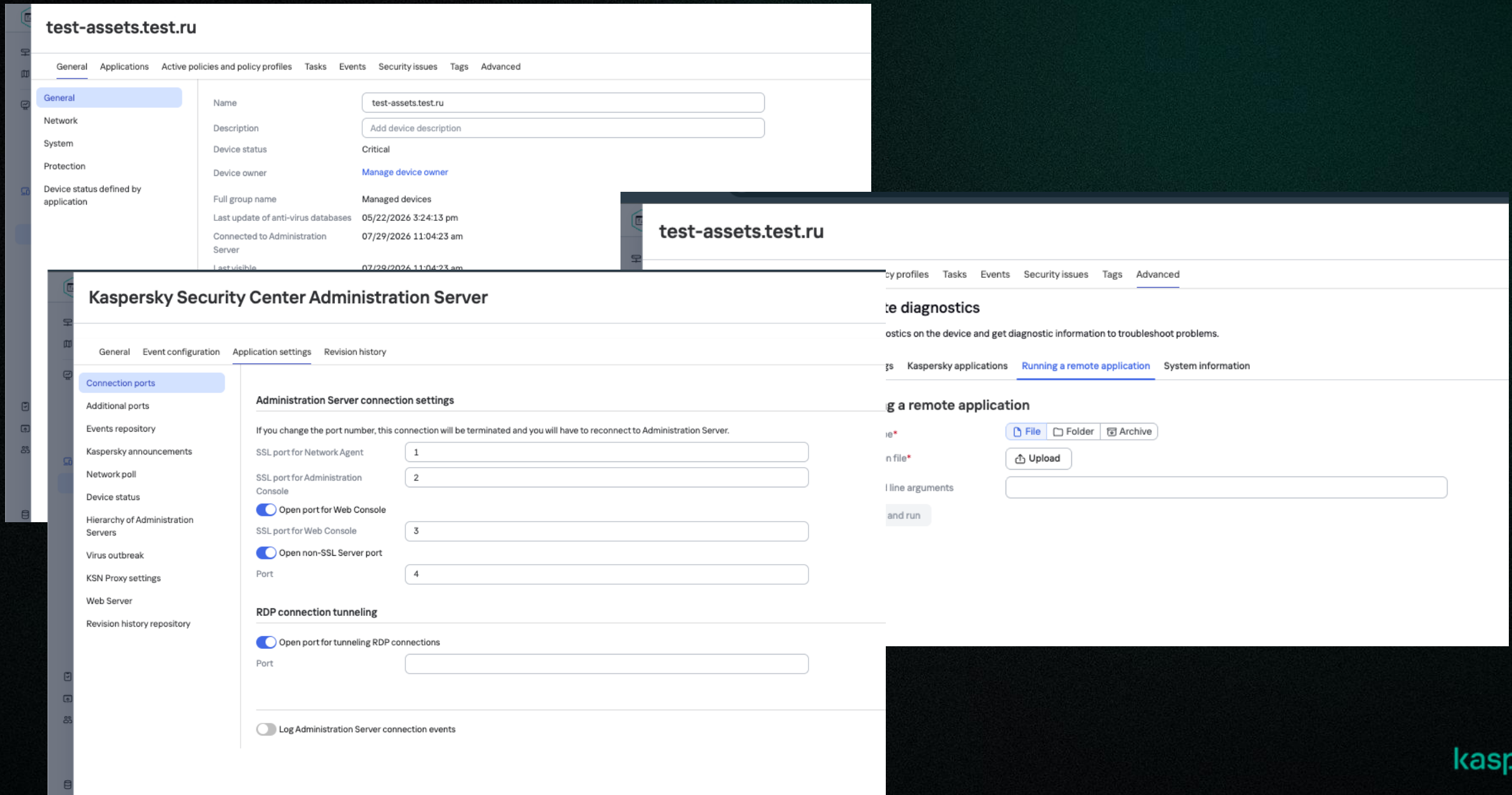

Kaspersky Security Center
kscvm-ui-01.avp.ru
Console navigation
Monitoring
Dashboard
Reports
Event selections
Asset management
Policies
Managed devices
Distribution points
Moving rules
Device selections
Tags
Unassigned devices
test-ui-01.test.ru
Managed devices
Policies
Hide hierarchy
test-ui-01.test.ru
Add
Refresh
Show in group
Copy
Move
Delete
Export
Import
Distribution
Status
Policy
Kaspersky Endpoint Security for Mac (12.2.1)
Kaspersky Endpoint Security for M
Kaspersky Security Center Administration Serve
Kaspersky Security Center Admini
Kaspersky Security Center Network Agent
Kaspersky Security Center Networ
Kaspersky Security Center Networ
Total 4 / Selected 0
Reports
Add
Open report template properties
Create delivery task
Export report
Delete
Refresh
Name
Type
Scope
Description
Protection status
Report on errors
Report on protection status
Deployment
Report on Kaspersky application versions
Report on incompatible applications
Report on license key usage by virtual Administ
Report on protection deployment
Report on usage of license keys
Updating
Managed devices
Add devices
Run task
Name
Last connected t...
Network Ag...
Status
Network...
Applicat...
IP addre...
Full gro...
OS packa
demo-vulns-asse
07/29/2026 11:04
No
Critical
16.3.0.707
10.80.82.50
Managed
apt
kscvm-ui-01
09/10/2026 3:00:3
Yes
Critical
16.3.0.823
10.80.82.48
Managed
apt
Total 2 / Selected 0

kaspersky


test-assets.test.ru
General
Applications
Active policies and policy profiles
Tasks
Events
Security issues
Tags
Advanced
General
Network
System
Protection
Device status defined by application
Name
test-assets.test.ru
Description
Add device description
Device status
Critical
Device owner
Manage device owner
Full group name
Managed devices
Last update of anti-virus databases
05/22/2026 3:24:13 pm
Connected to Administration Server
07/29/2026 11:04:23 am
Last visible
07/29/2026 11:04:23 am
Network Agent version
16.3.0.707
Created
06/06/2026 12:00:10 pm
Force synchronization
Do not disconnect from the Administration Server
Sessions
User name
User principal name
SAM
demo-vulns-asset-lin-2.avp.ru\root
root

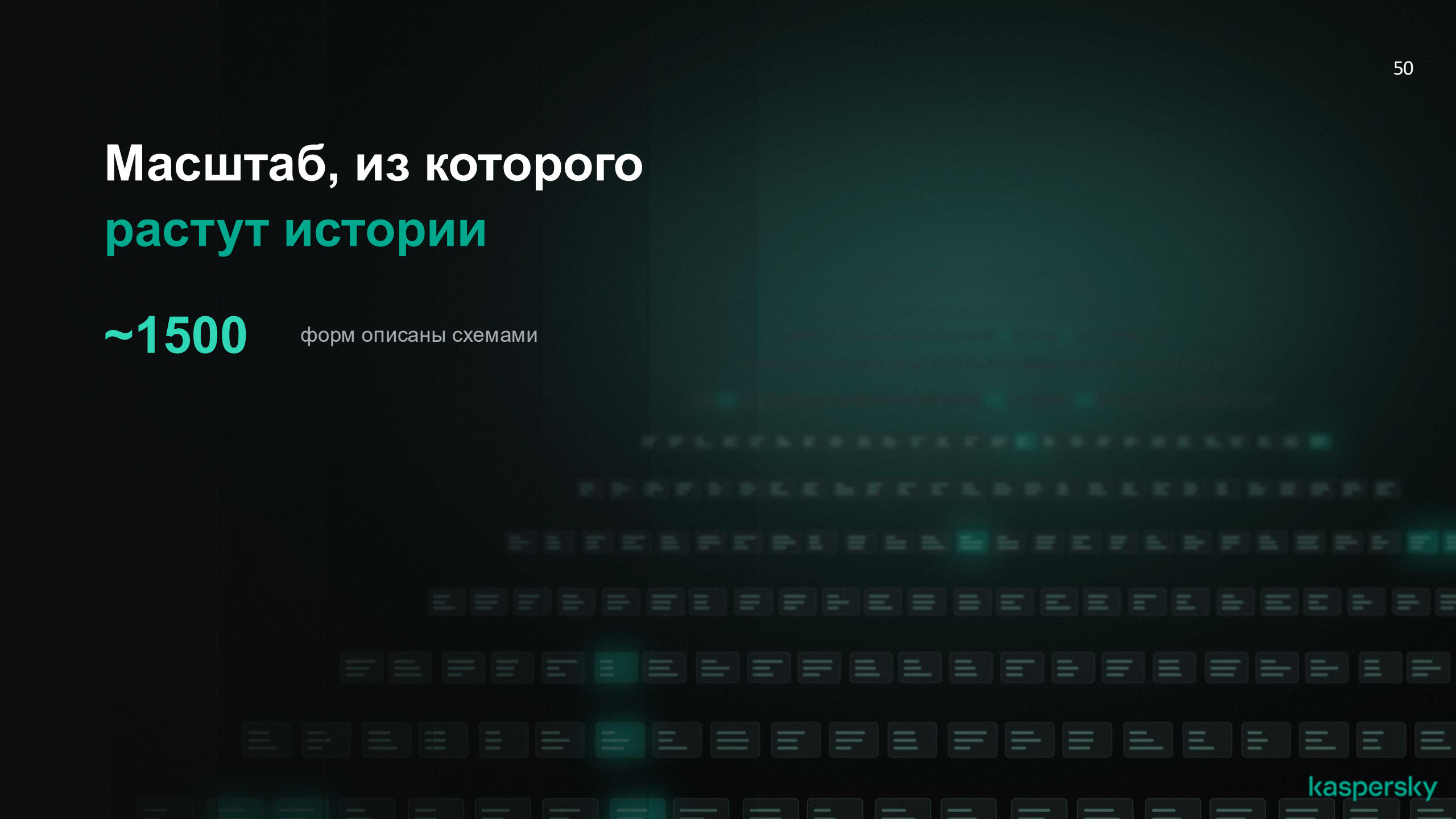

test-assets.test.ru
General
Applications
Active policies and policy profiles
Tasks
Events
Security issues
Tags
Advanced
General
Network
System
Protection
Device status defined by application
Name
test-assets.test.ru
Description
Add device description
Device status
Critical
Device owner
Manage device owner
Full group name
Managed devices
Last update of anti-virus databases
05/22/2026 3:24:13 pm
Connected to Administration Server
07/29/2026 11:04:23 am
Last visible
07/29/2026 11:04:23 am
Network Agent version
16.3.0.707
Created
06/06/2026 12:00:10 pm
Force synchronization
Do not disconnect from the Administration Server
Sessions
User name
demo-vulns-asset-lin-2.avp.ru\root


test-assets.test.ru
General
Applications
Active policies and policy profiles
Tasks
Events
Security issues
Tags
Advanced
Executable files
Distribution points
Hardware registry
Available updates
Remote diagnostics
Seamless updates
Remote diagnostics
Run diagnostics on the device and get diagnostic information to troubleshoot problems.
Event logs
Kaspersky applications
Running a remote application
System information
Running a remote application
Object type*
File
Folder
Archive
Application file*
Upload
Command line arguments
Upload and run


test-assets.test.ru
General
Applications
Active policies and policy profiles
Tasks
Events
Security issues
Tags
Advanced
General
Network
System
Protection
Device status defined by application
Name
test-assets.test.ru
Description
Add device description
Device status
Critical
Device owner
Manage device owner
Full group name
Managed devices
Last update of anti-virus databases
05/22/2026 3:24:13 pm
Connected to Administration Server
07/29/2026 11:04:23 am
Kaspersky Security Center Administration Server
General
Event configuration
Application settings
Revision history
Connection ports
Additional ports
Events repository
Kaspersky announcements
Network poll
Device status
Hierarchy of Administration Servers
Virus outbreak
KSN Proxy settings
Web Server
Revision history repository
Administration Server connection settings
If you change the port number, this connection will be terminated and you will have to reconnect to Administration Server.
SSL port for Network Agent
1
SSL port for Administration Console
2
Open port for Web Console
SSL port for Web Console
3
Open non-SSL Server port
Port
4
RDP connection tunneling
Open port for tunneling RDP connections
Port
Log Administration Server connection events
test-assets.test.ru
Tasks
Events
Security issues
Tags
Advanced
Kaspersky applications
Running a remote application
System information
File
Folder
Archive
Upload
and run

Масштаб, из которого растут истории

~1500 форм описаны схемами

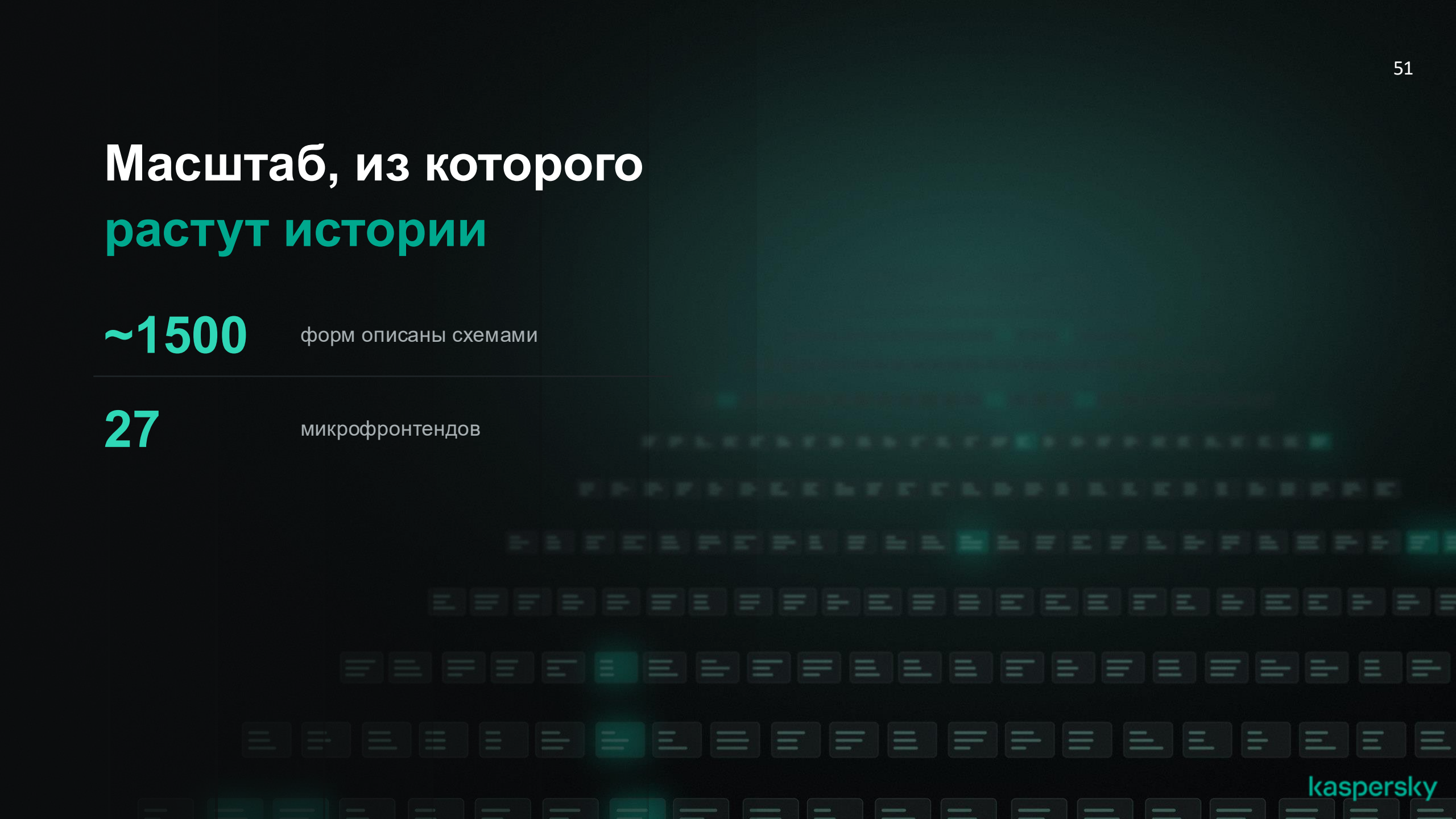


kaspersky

Масштаб, из которого растут истории

~1500 форм описаны схемами

27 микрофронтендов



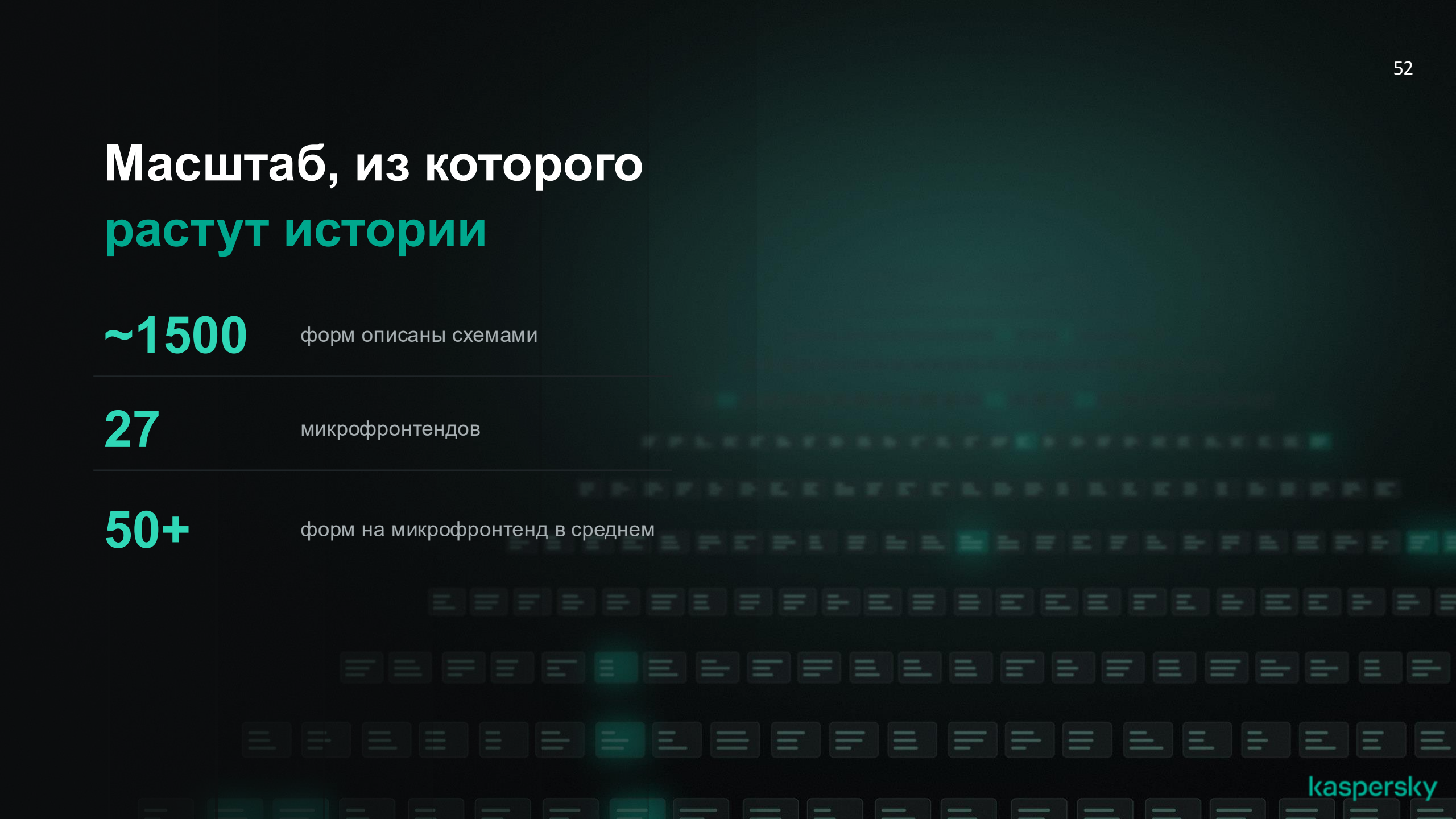
kaspersky

Масштаб, из которого растут истории

~1500 форм описаны схемами

27 микрофронтендов

50+ форм на микрофронтенд в среднем



Масштаб, из которого растут истории

~1500 форм описаны схемами

27 микрофронтендов

50+ форм на микрофронтенд в среднем

704

формы в одном микрофронтенде

kaspersky

Секундочку, я записываю…



А если кодить руками?

Армия фронтендеров

Работа конвейерная, но ручная. Каждая новая форма - ещё один человек-неделя, и так сотни раз.



А если кодить руками?

Потеря контроля

Своя вёрстка, своя валидация, своё понимание отступов в каждой команде. Баги плодятся копипастой.



А если делать правильно

**Интерфейс не через код,
а через описание.**



Backend Driven UI

- Схема приезжает с сервера на лету
- Можно менять экран без релиза клиента
- Ревью схемы - на стороне сервера

kaspersky

Backend Driven UI

- Схема приезжает с сервера на лету
- Можно менять экран без релиза клиента
- Ревью схемы - на стороне сервера

Schema Driven UI

- Схема живёт в репозитории микрофронтенда
- Собирается в артефакт вместе с продуктом
- Проходит обычное код-ревью



UI-as-config

Интерфейс как конфигурация

UI-as-config

Интерфейс как конфигурация

Наш случай: Schema Driven UI

Репозиторий > Ревью > Сборка > Пользователь

схема лежит в git, проходит ревью в том же pull request и едет к пользователю внутри пакета



UI-as-config

Интерфейс как конфигурация

Наш случай: Schema Driven UI

Репозиторий > Ревью > Сборка > Пользователь

схема лежит в git, проходит ревью в том же pull request и едет к пользователю внутри пакета

Классический BDUI

Сервер > Браузер

схема приезжает по сети каждый раз, когда пользователь открывает экран

Часть 3

Бизнес и инженерка
Плюсы подхода



Что это даёт бизнесу?

Скорость

однотипные формы собираются быстрее

Что это даёт бизнесу?

Скорость

однотипные формы собираются быстрее

Меньше ручного кода

разработчик описывает форму, а не верстает

Что это даёт бизнесу?

Скорость

однотипные формы собираются быстрее

Меньше ручного кода

разработчик описывает форму, а не верстает

Переиспользование

компоненты общие, консистентность

Что это даёт бизнесу?

Скорость

однотипные формы собираются быстрее

Меньше ручного кода

разработчик описывает форму, а не верстает

Переиспользование

компоненты общие, консистентность

Визуальный конструктор

собрал мышкой - получил схему

Что это даёт бизнесу?

Скорость

однотипные формы собираются быстрее

Меньше ручного кода

разработчик описывает форму, а не верстает

Переиспользование

компоненты общие, консистентность

Визуальный конструктор

собрал мышкой - получил схему

Быстрые изменения

правка конфигурации, а не компонента

Что это даёт бизнесу?

Скорость

однотипные формы собираются быстрее

Меньше ручного кода

разработчик описывает форму, а не верстает

Переиспользование

компоненты общие, консистентность

Визуальный конструктор

собрал мышкой - получил схему

Быстрые изменения

правка конфигурации, а не компонента

Проще эксперименты

варианты экрана - просто разные схемы

Почему это нравится инженерам

+ Единый механизм отрисовки



Почему это нравится инженерам

+ Единый механизм отрисовки

+ Общий формат описания



Почему это нравится инженерам

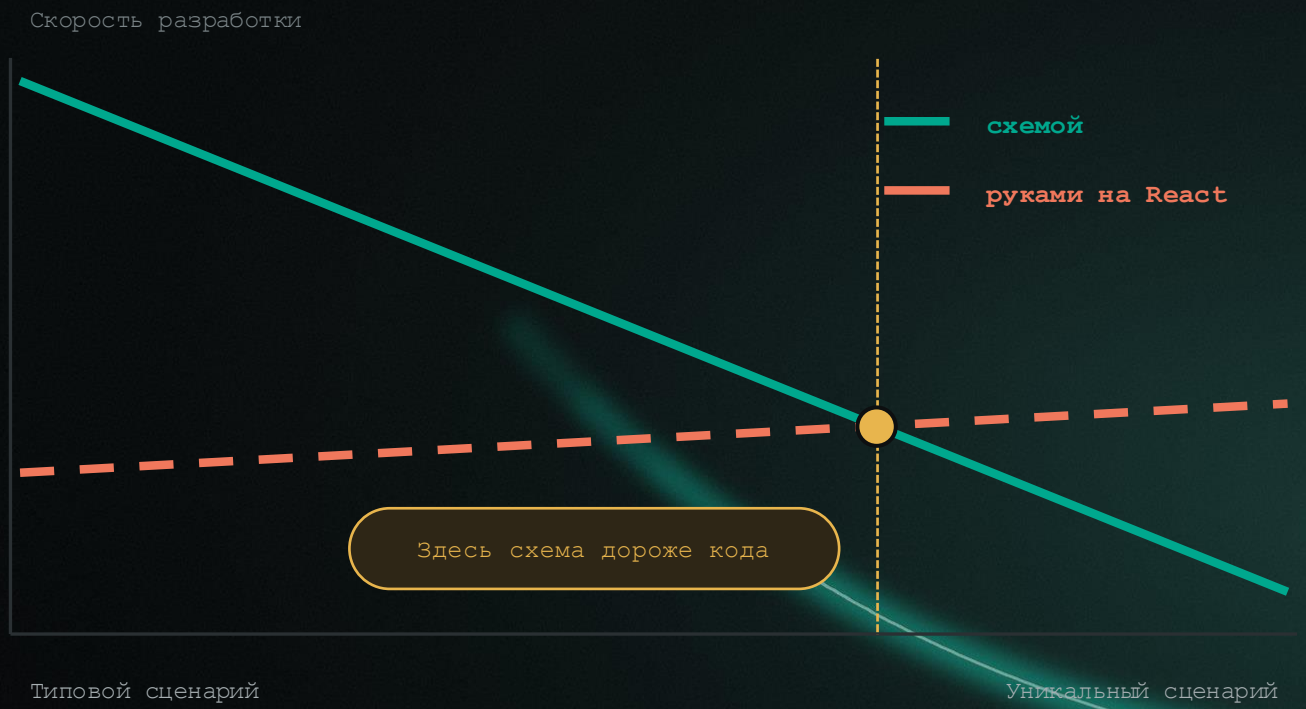
+ Единый механизм отрисовки
+ Общий формат описания
+ Единая модель валидации



Почему это нравится инженерам

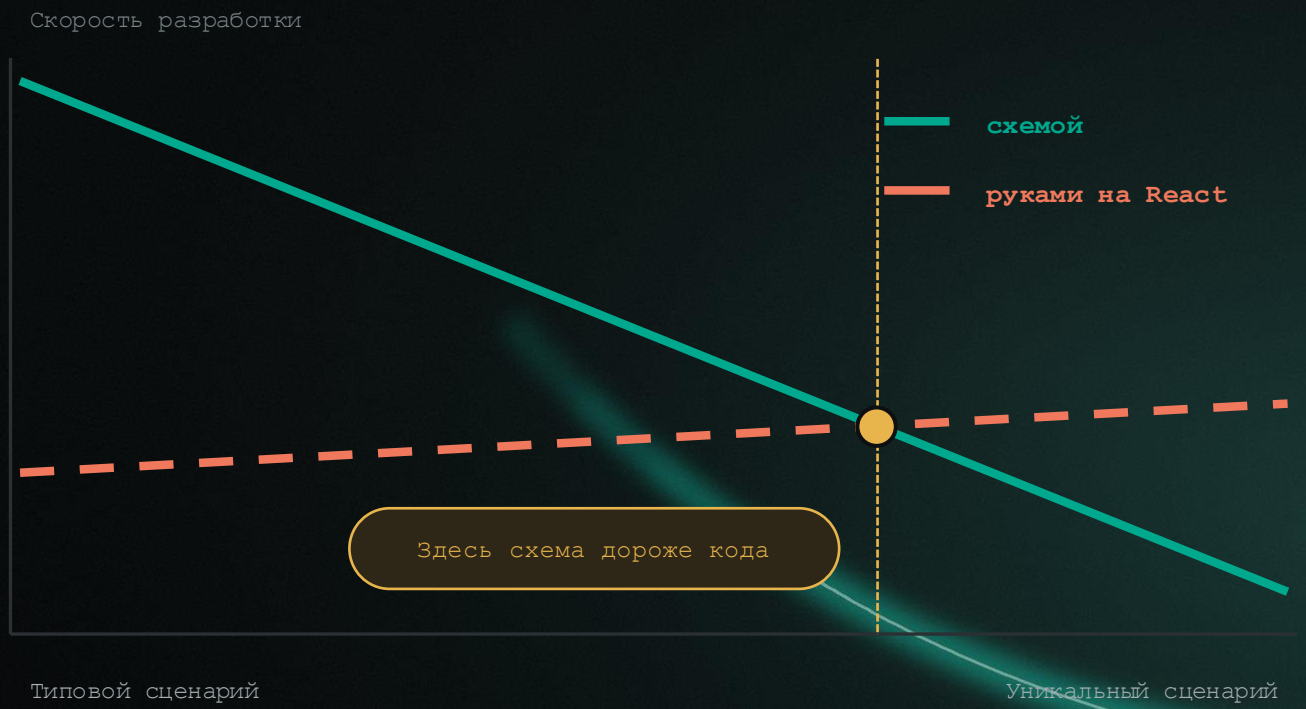
+ Единый механизм отрисовки
+ Общий формат описания
+ Единая модель валидации
+ Меньше копипасты





Вот тут важно

Скорость появляется только там, где сценарии хорошо укладываются в DSL. Как только сценарий уникален, схема становится дороже ручного React.



Вот тут важно

Скорость появляется только там, где сценарии хорошо укладываются в DSL. Как только сценарий уникален, схема становится дороже ручного React.

Это не «мы больше не пишем фронтенд». Это «мы не пишем фронтенд для типовых сценариев».

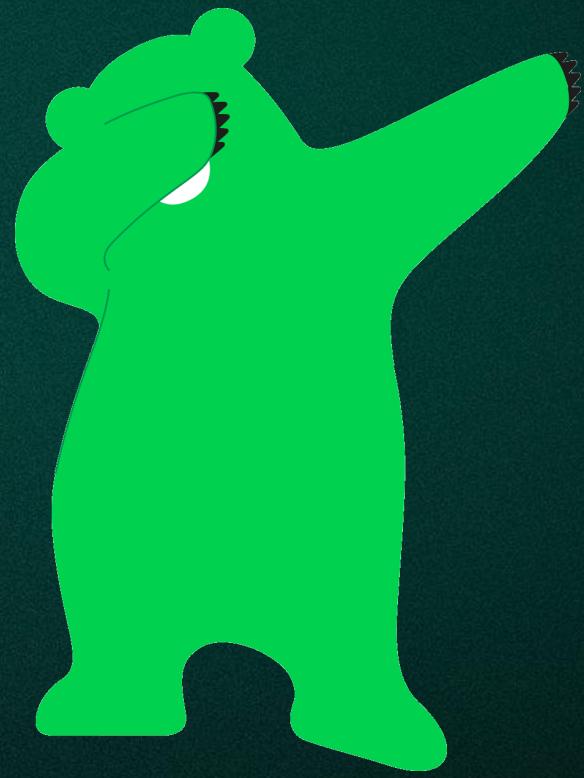
Часть 4

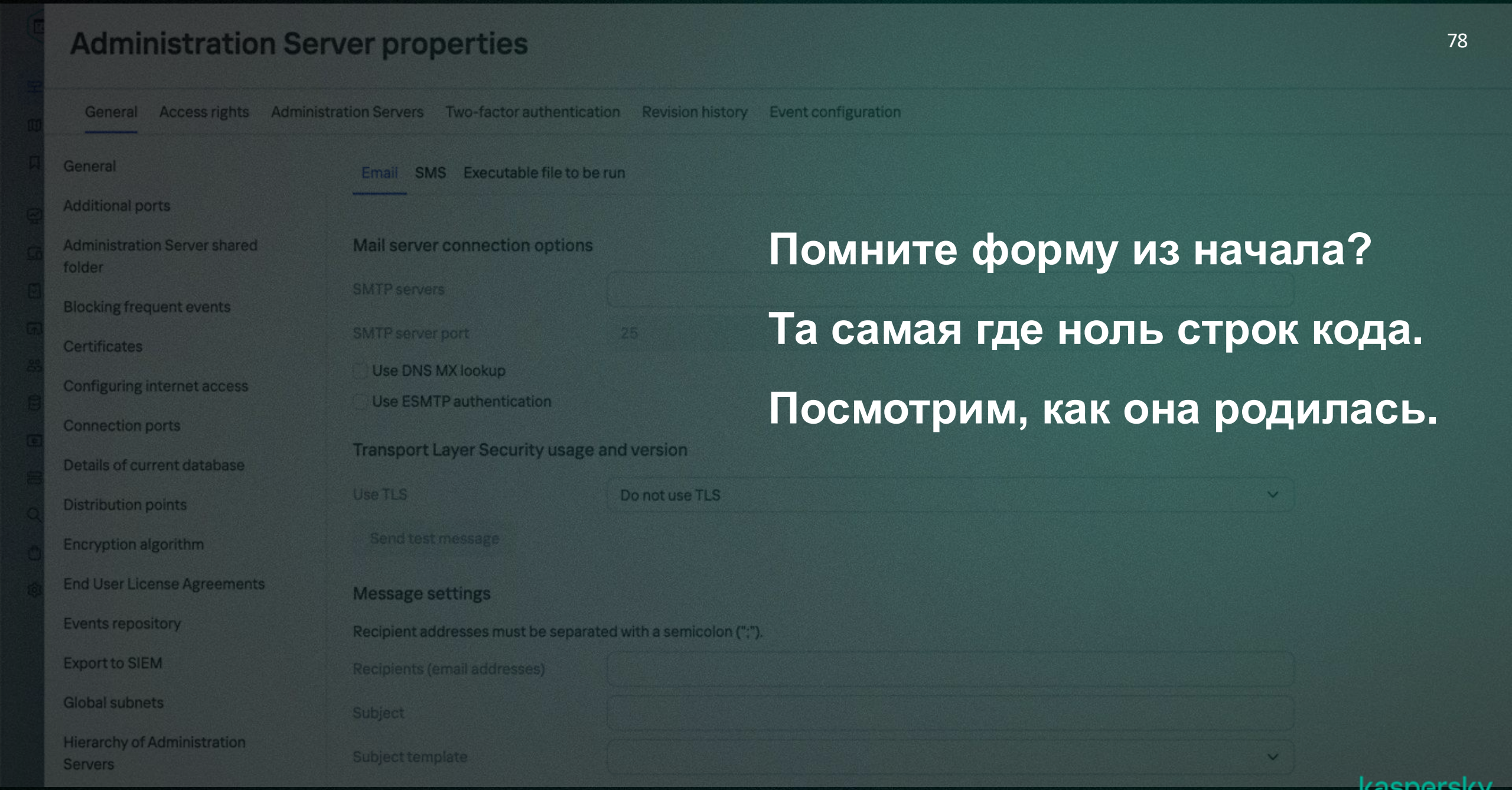
DSL, движок, миграции
Как это устроено



Привет, форма!

Глава первая





Помните форму из начала?

Та самая где ноль строк кода.

Посмотрим, как она родилась.

От «придумали» до «увидел пользователь»

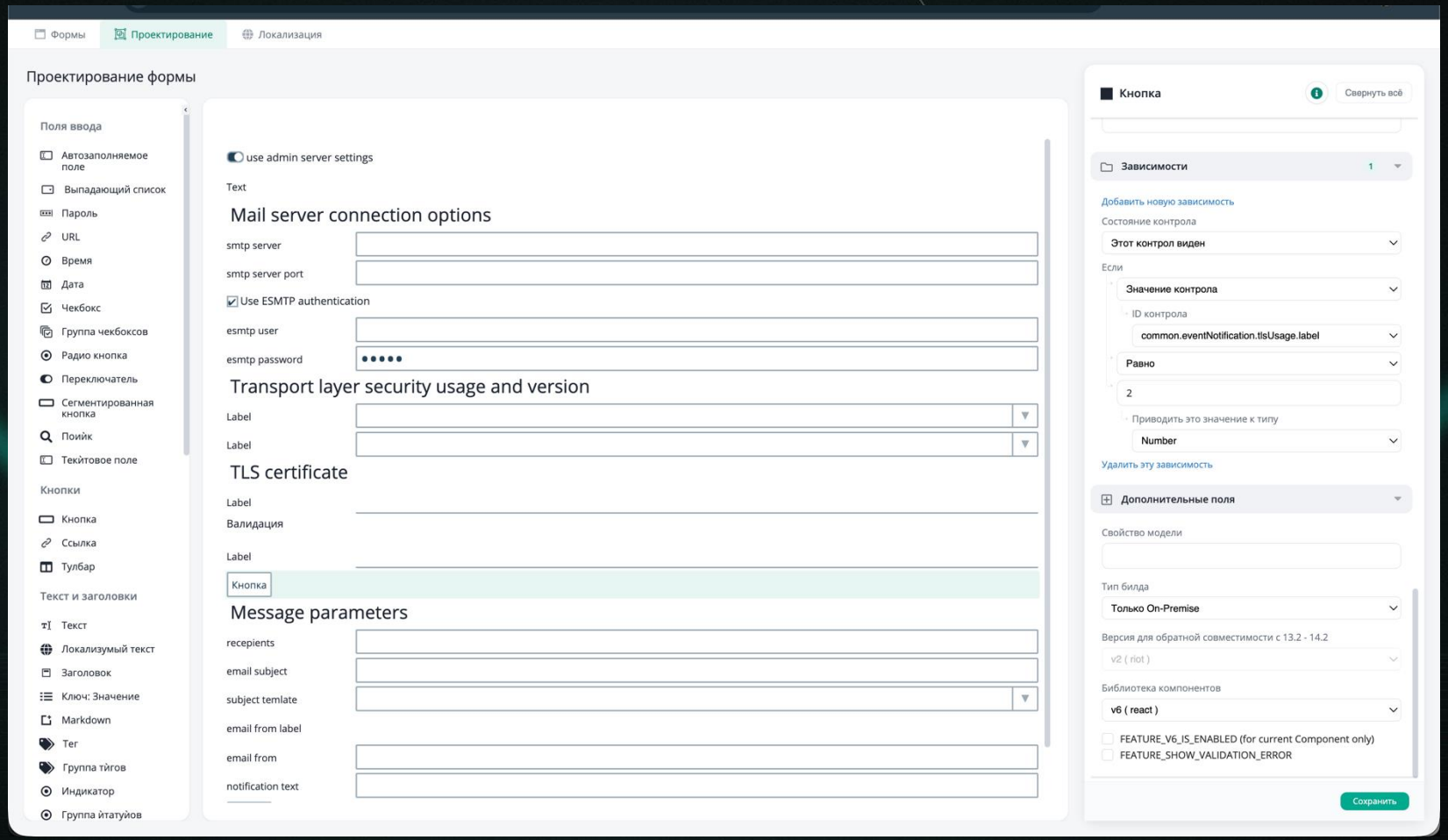
Конструктор

визуальный редактор

–
>

JSON-схема

в репозитории



От «придумали» до «увидел пользователь»

Конструктор

визуальный редактор

–>

JSON-схема

в репозитории

Сборка

buildUi + jsonpack

–>

Реестр

Регистрация формы в сторе

Build-ui.js

сама утилита buildUi. Проходит по всем сырым схемам из UI Builder и выкидывает служебные данные собирая форму по белому списку полей

```
function buildUi ({ forms, ...}) {
 const processedForms = {}

 for (const formId in forms) {
  // из схемы получаем только белый список:
  processedForms[id] = {
   id, name, elements, /* ... */
  }
 }
 return processedForms
}
```

От «придумали» до «увидел пользователь»

Конструктор

визуальный редактор

->

JSON-схема

в репозитории

Сборка

buildUi + jsonpack

->

Реестр

Регистрация формы в сторе

forms.js

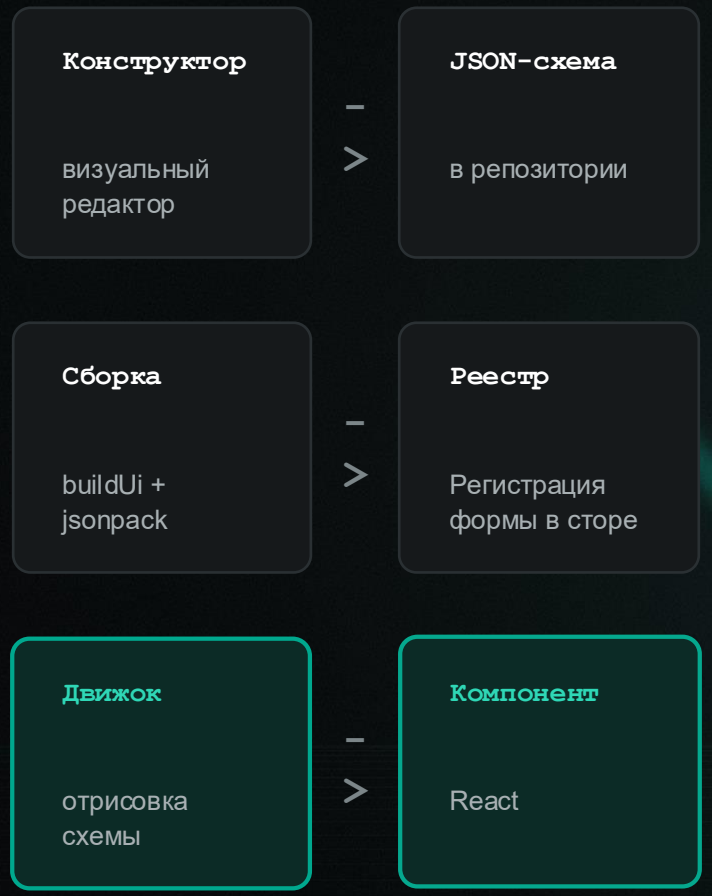
Регистрация формы в реестре

```
function FormsService (runtimeInstance) {
 const formsStore = {}

  function registerForm (form) {
   formsStore[form.id] = schemePatcher(form, formsStore)
  }

  function getForm (formId) {
   return formsStore[formId]
  }
/* ... */
}
```

От «придумали» до «увидел пользователь»



build.js

Откат вместо падения: если адаптера под новую версию нет, движок молча переключает компонент на проверенную версию

```
async function build (uiElement) {
 // 1. решаем, какая версия компонента нужна
 const { type, version } = switchLib(uiElement)

 if (isReactLib(version)) {
  const adapter = await checkComponentAdapter({
   type: type,
   libVersion: version
  })

 if (adapter.isExists) {
  // собираем react-версию компонента ...
 } else {
  // адаптера нет - откат на проверенную версию
  uiElement.libVersion = ‘ stable-lib’
 }
}
```

Три идеи, без которых дальше никак

1

**Схема - данные,
а не код**

JSON нельзя выполнить, его можно только интерпретировать. Всё поведение живёт в коде.

Три идеи, без которых дальше никак

1

Схема - данные, а не код

JSON нельзя выполнить, его можно только интерпретировать. Всё поведение живёт в коде.

2

Тип элемента -> компонент из реестра

Мапа - движок видит тип и спрашивает, кто умеет это рисовать.

Три идеи, без которых дальше никак

1

Схема - данные, а не код

JSON нельзя выполнить, его можно только интерпретировать. Всё поведение живёт в коде.

2

Тип элемента -> компонент из реестра

Мапа - движок видит тип и спрашивает, кто умеет это рисовать.

3

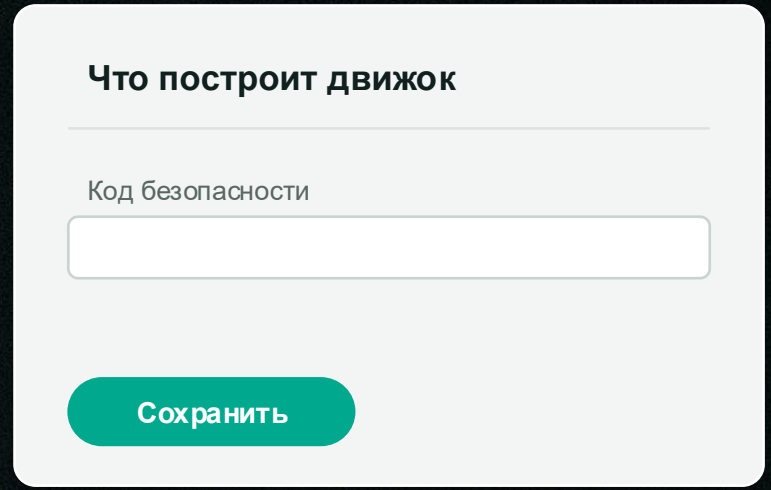
Модель данных - общая на всю форму

Каждый контрол знает, к какому куску модели он привязан.

Как выглядит схема на самом деле

2fa-code.json

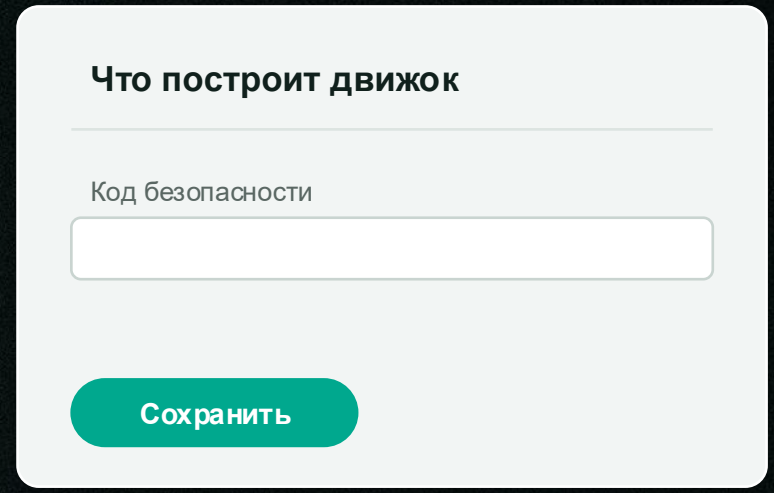
```
{
  "type": "password-input",
  "id": "6NJ1tqrnk",
  "title": "2FA.securityCode.enterSecurityCode",
  "required": true,
  "value": {
    "mapTo": { "type": "property", "name": "securityCode" }
  },
  "dependencies": [],
  "validation": {
    "rules": [
      {
        "source": {
          "command": "handlers.2fa.codeValidation"
        }
      }
    ]
  },
  "componentLibrary": "riot",
  "componentLibraryNext": "v6"
}
```



Как выглядит схема на самом деле

2fa-code.json

```
{
 "type": "password-input",
 "id": "6NJ1tqrnk",
 "title": "2FA.securityCode.enterSecurityCode",
 "required": true,
 "value": {
  "mapTo": { "type": "property", "name": "securityCode" }
 },
 "dependencies": [],
 "validation": {
  "rules": [
   {
    "source": {
     "command": "handlers.2fa.codeValidation"
    }
   }
  ]
 },
 "componentLibrary": "riot",
 "componentLibraryNext": "v6"
}
```



Что описано декларативно

`type`	тип компонента

Что описано декларативно

`type`	тип компонента
`value.mapTo`	куда писать значение

Что описано декларативно

`type`	тип компонента
`value.mapTo`	куда писать значение
`title`	не текст, а ключ локализации

Что описано декларативно

`type`	тип компонента
`value.mapTo`	куда писать значение
`title`	не текст, а ключ локализации
`validation`	правила проверки - по имени функции из кода микрофронтенда

Что описано декларативно

`type`	тип компонента
`value.mapTo`	куда писать значение
`title`	не текст, а ключ локализации
`validation`	правила проверки - по имени функции из кода микрофронтенда
`dependencies`	«покажи это поле, только если вот то равно вот этому»

Что описано декларативно

`type`	тип компонента
`value.mapTo`	куда писать значение
`title`	не текст, а ключ локализации
`validation`	правила проверки - по имени функции из кода микрофронтенда
`dependencies`	«покажи это поле, только если вот то равно вот этому»
`componentLibrary`	выбор библиотеки компонентов - наша миграция UI kit

«Просто JSON» оказался маленьким языком программирования.

Переменные

привязки к данным

«Просто JSON» оказался маленьким языком программирования.

Переменные

привязки к данным

Условия

зависимости между полями

hierarchy.json

```
"dependencies": [{
 "set": { "state": "visible" },
 "when":{
       "varName":   ”everythingIsOk",
       "value": true
      }
}]
```

«Просто JSON» оказался маленьким языком программирования.

Переменные

привязки к данным

Условия

зависимости между полями

Функции

обработчики и правила по имени

2fa-code.json

```
{
 "type": "password-input",
 "id": "6NJ1tqrnk",
 "title": "2FA.securityCode.enterSecurityCode",
 "required": true,
 "value": {
  "mapTo": { "type": "property", "name": "securityCode" }
 },
 "dependencies": [],
 "validation": {
  "rules": [
   {
    "source": {
     "command": "handlers.2fa.codeValidation"
    }
   }
  ]
 },
 "componentLibrary": "riot",
 "componentLibraryNext": "v6"
}
```

«Просто JSON» оказался маленьким языком программирования.

Переменные

привязки к данным

Условия

зависимости между полями

Функции

обработчики и правила по имени

2fa-security-code.ts

```
export const validateCode = () => {
const model = form.getModel();

 return {
  valid:!model.hasCodeError,
  errors: model.hasCodeError ? [{ message: ...}] : [],
  };
 };

runtime.join("handlers.2fa.codeValidation", validateCode);
```

«Просто JSON» оказался маленьким языком программирования.

Переменные

привязки к данным

Условия

зависимости между полями

Функции

обработчики и правила по имени

Миграции

режимы совместимости версий

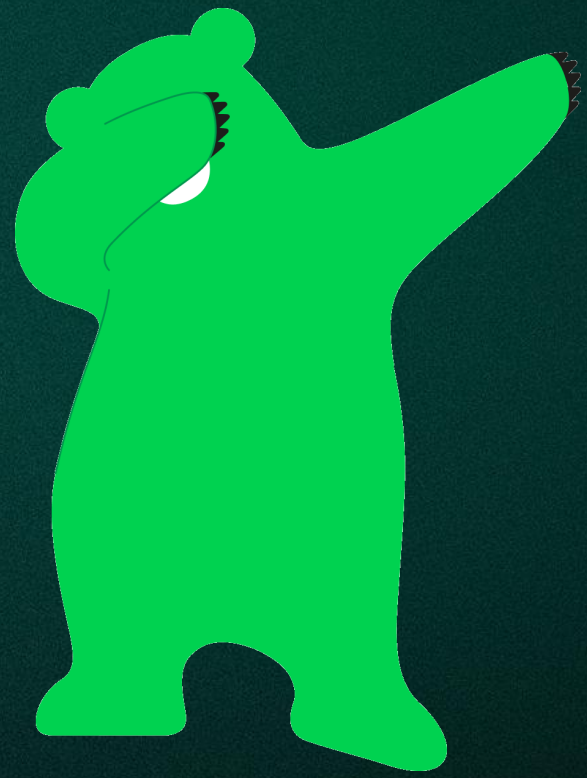


Как любой язык, DSL требует грамматики, версионирования и правил эволюции.



Существование формы

Глава вторая



Применить

одна невинная кнопка

…за которой тянется дремучий процесс

Сходить в бекенд

Применить

…за которой тянется дремучий процесс

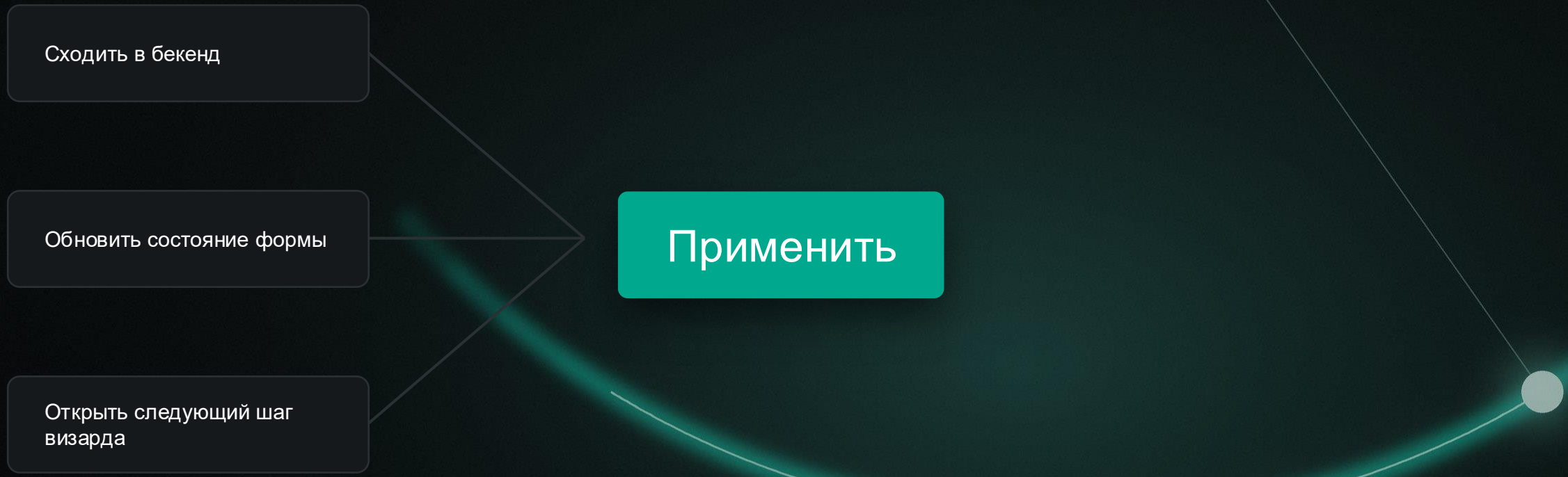
Сходить в бекенд

Обновить состояние формы

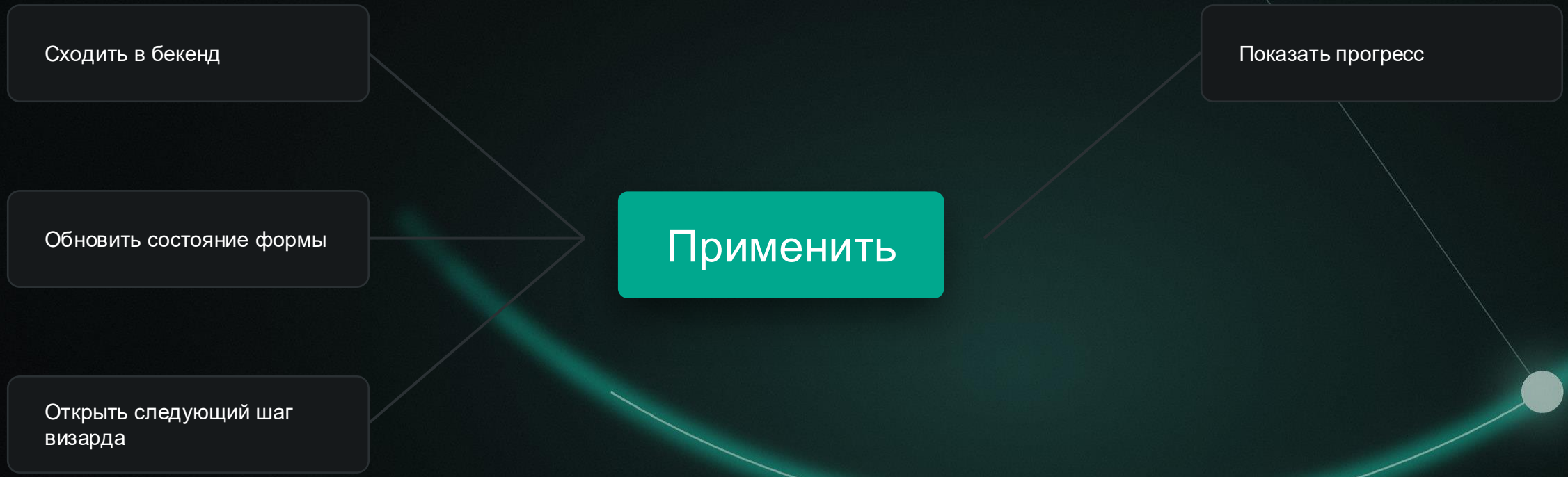
Применить

kaspersky

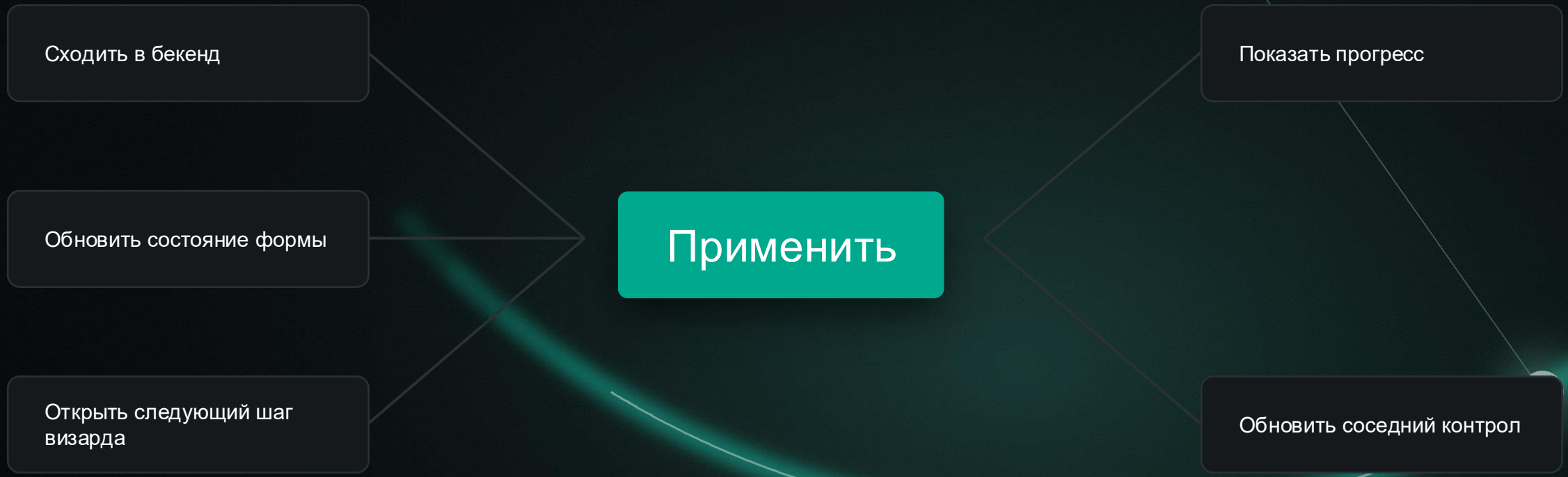
…за которой тянется дремучий процесс



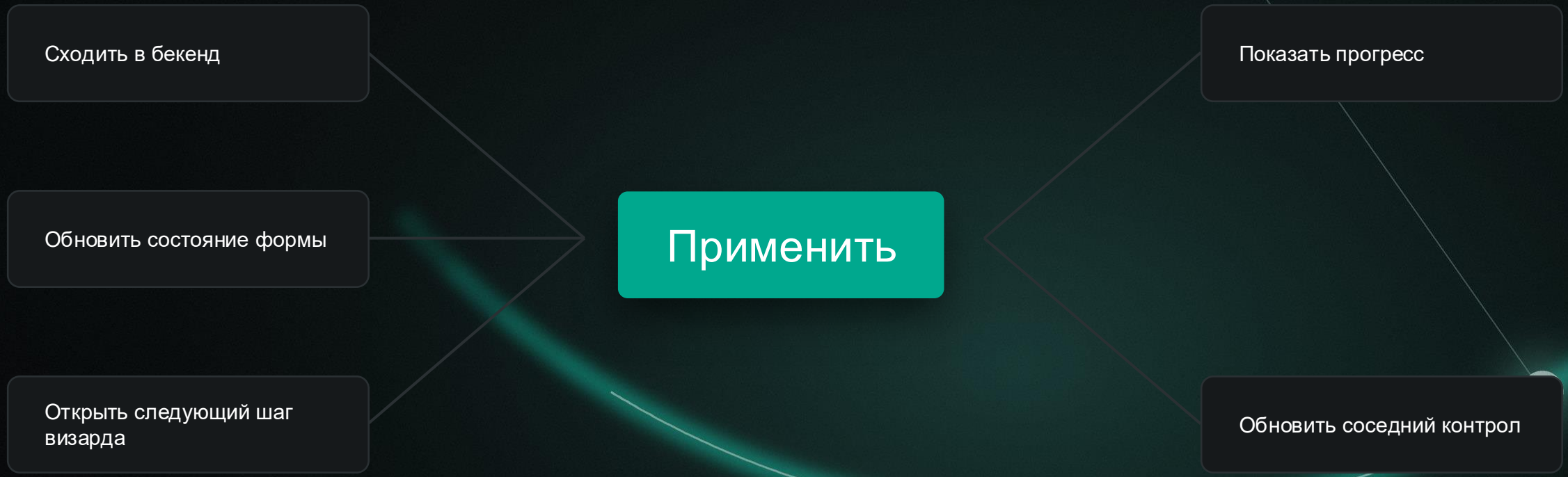
…за которой тянется дремучий процесс



…за которой тянется дремучий процесс



…за которой тянется дремучий процесс



Форма в enterprise почти всегда не ограничивается простым набор полей.

Два одновременно плохих решения

Всё в DSL

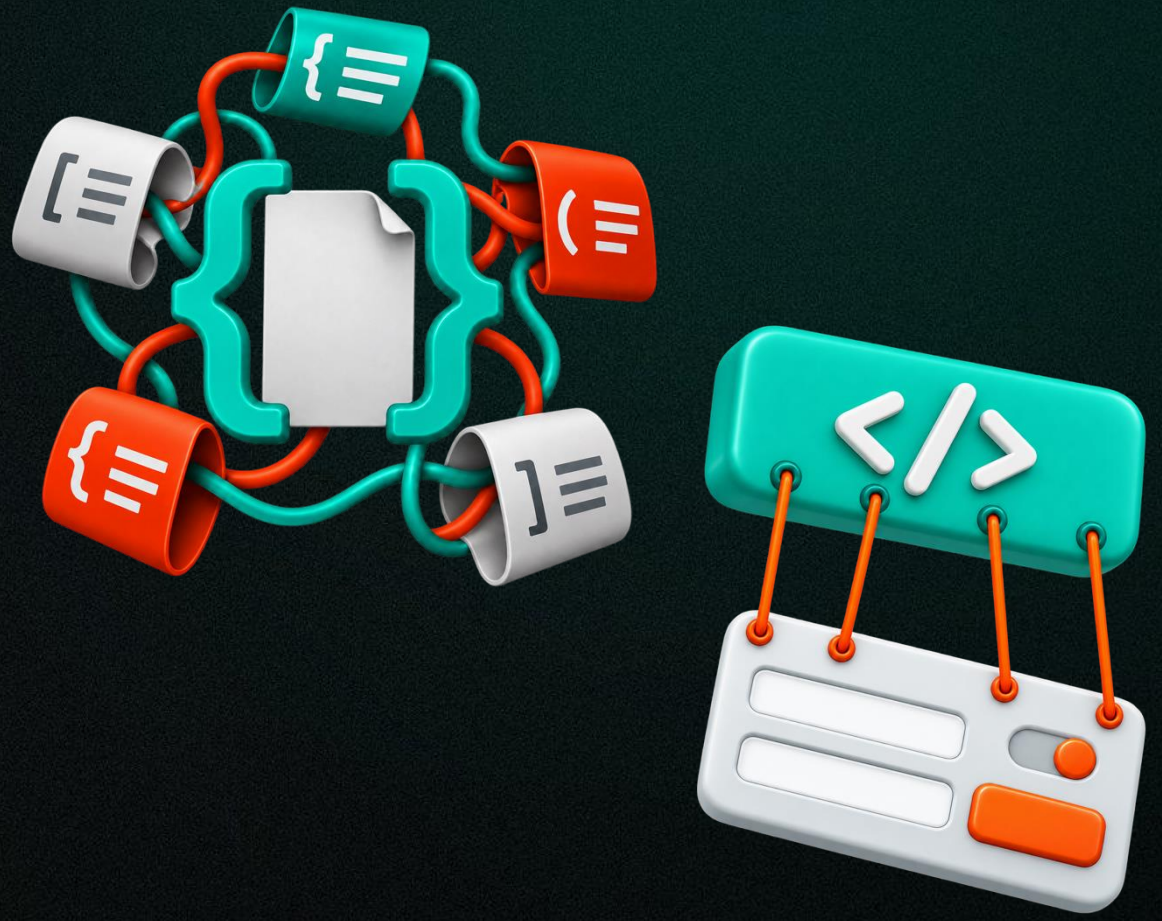
Схема превращается в нечитабельную программу на три тысячи строк - на языке, у которого нет ни отладчика, ни типов.



Два одновременно плохих решения

Всё в коде

Смотришь на схему - и не понимаешь, что форма вообще делает. Описание перестаёт быть описанием.



В DSL - ЧТО. В коде - КАК.

settings.json (схема)

```
"dataSource": {
  "dsType": "function",
  "staticFunctionName": "handlers.settings.getData",
  "serverFunctionName": "getFormModel"
},
"init": { "fncName": "handlers.settings.activate" }
```



В DSL - ЧТО. В коде - КАК.

settings.json (схема)

```
"dataSource": {
  "dsType": "function",
  "staticFunctionName": "handlers.settings.getData",
  "serverFunctionName": "getFormModel"
},
"init": { "fncName": "handlers.settings.activate" }
```

->

по имени

settings-get-model.ts (код)

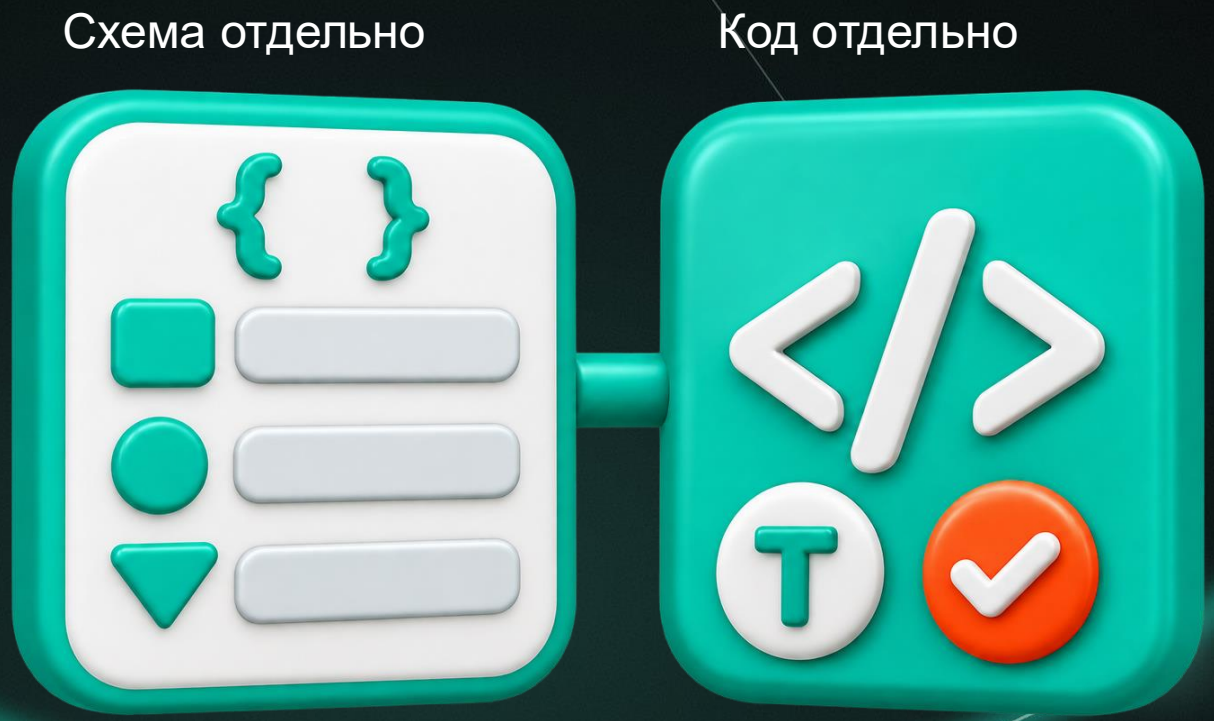
```
const settingsGetModel = async ({ result }) => {
  const settings = await settingsModel.getSettings(id)
  result({…})
}

runtime.join('handlers.settings.getData', …)
```



Что хорошо

- Схема остаётся читаемой
- Логика - обычный тестируемый код с типами
- DSL не превращается в язык с циклами



Что плохо

- Строку нельзя кликнуть в IDE
- Опечатку не ловит компилятор - ловит пользователь
- Рефакторинг переименованием опасен

settings.json (схема)

```
"dataSource": {
  "dsType": "function",
  "staticFunctionName": "handlers.settings.getData",
  "serverFunctionName": "getFormModel"
},
"init": { "fncName": "handlers.settings.activate" }
```

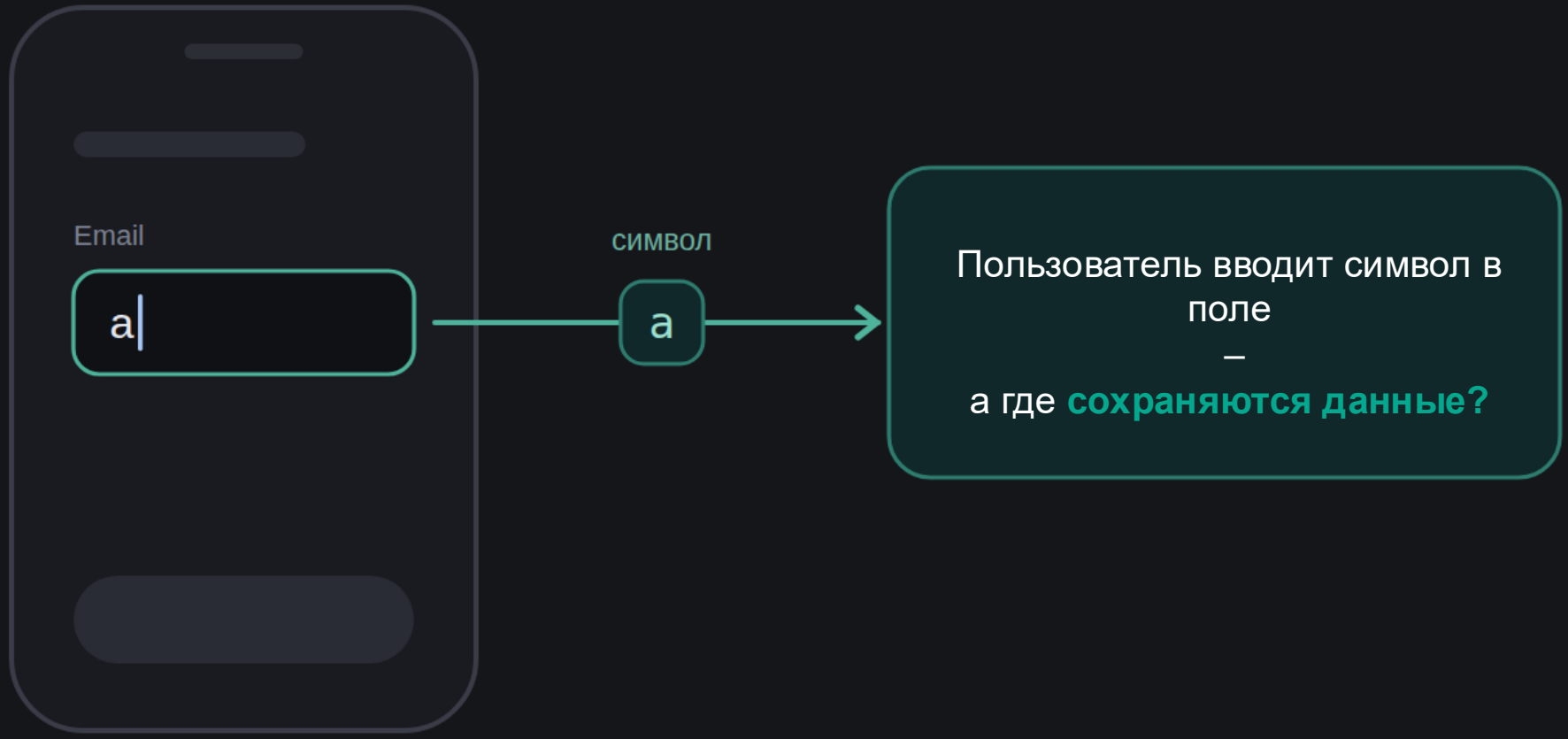


Делаем ссылки по именам - сразу пишем проверку, что каждое имя из каждой схемы существует в коде.

Такие проверки **AI** пишет отлично - работа механическая.



Где живут данные формы?



Где живут данные формы

dataSource

начальные данные из схемы

В DSL - ЧТО. В коде - КАК.

settings.json (схема)

```
"dataSource": {
  "dsType": "function",
  "staticFunctionName": "handlers.settings.getData",
  "serverFunctionName": "getFormModel"
},
"init": { "fncName": "handlers.settings.activate" }
```

Где живут данные формы

dataSource
начальные данные из схемы

↓

ModelProvider
общее хранилище формы, новая версия на каждое изменение

В DSL - ЧТО. В коде - КАК.

settings.json (схема)

```
"dataSource": {
  "dsType": "function",
  "staticFunctionName": "handlers.settings.getData",
  "serverFunctionName": "getFormModel"
},
"init": { "fncName": "handlers.settings.activate" }
```

Где живут данные формы

dataSource
начальные данные из схемы

ModelProvider
общее хранилище формы, новая версия на каждое изменение

Событие
«данные изменились»

Подписанные
перерисовываются, включая связанные поля

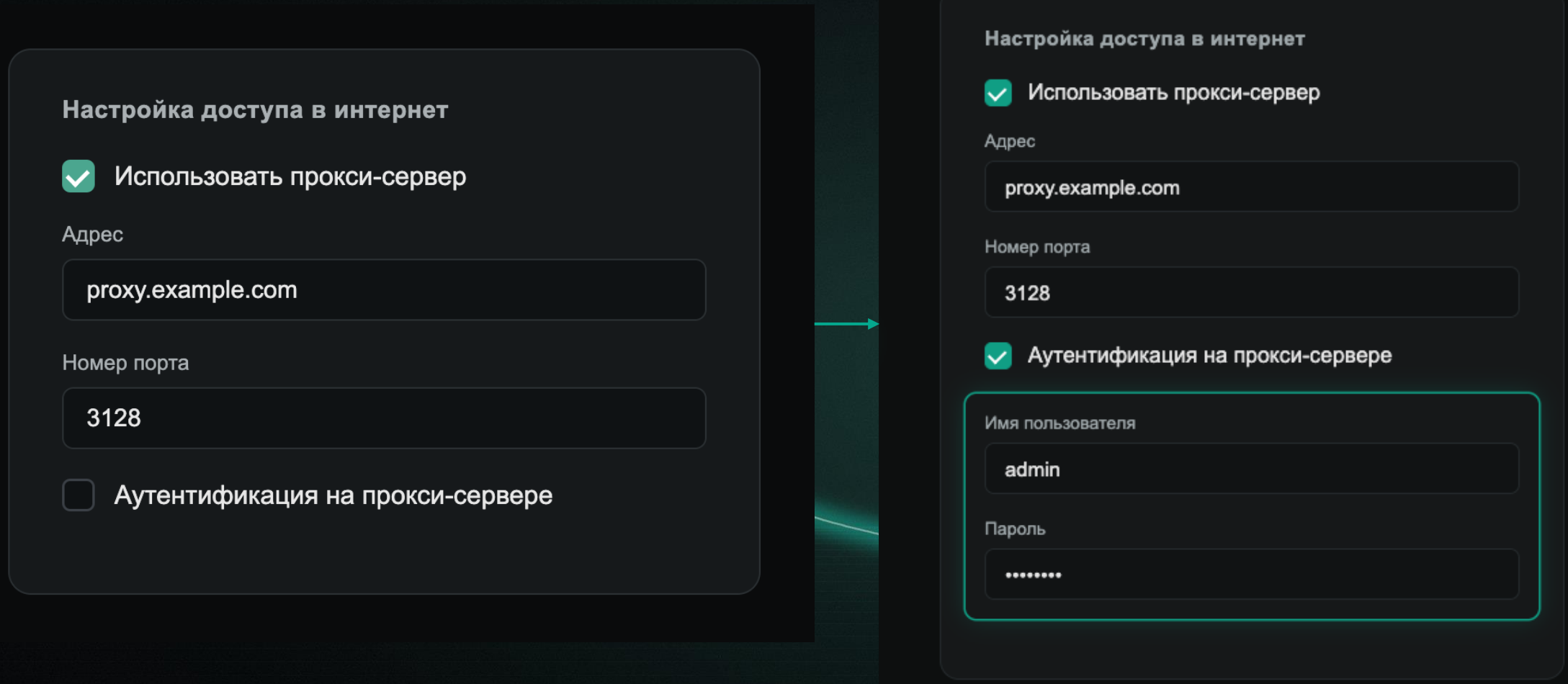
hierarchy.json

```
"dependencies": [{
 "set": { "state": "visible" },
 "when":{
    "varName":”everythingIsOk",
          "value": true
    }
}]
```

provider.js

```
scope[formId].set(ctx)
feed.emit(`form:${formId}:data:changed`, ctx)
feed.emit('data:changed', formId, ctx)
```

Где живут данные формы



react-hook-form - та же идея

Единое хранилище, поля по имени, подписки, флаги - всё это там уже есть

LoginForm.jsx (react-hook-form)

```
// хранилище
const { register, watch, formState } = useForm()

// поле по имени
<input {...register('host')} />

// подписка
watch('host')

// флаги
formState.isDirty   formState.isValid
```

form.json + kl-auto-ui (у нас)

```
// хранилище + история версий
ModelProvider

// поле по имени - в схеме
{ "type": "input", "id": "host" }

// подписка
feed.on('data:changed', handler)

// флаги
«отредактирована»   «невалидна»
```

Три уровня проверки данных

1 Встроенные правила

Целый отдельный пакет с валидацией

```
"validation": {
  "rules": [
    { "method": "notEmpty" },
    { "method": "integer" }
  ]
}
```

notEmpty range integer …

33 правила в библиотеке

2 Кастомные по имени

функция живёт в коде плагина, схема зовёт её по строке

```
// form.json
{ "method": "function",
  "functionName":
  "handlers.2fa.codeValidation" }
```

↓ runtime.getSync ↓

```
// код плагина
runtime.join(
  'handlers.2fa.codeValidation',
  validate2FACode)
```

3 Серверная проверка

при сохранении клиентской валидации никогда не достаточно

форма
↓
клиент +
↓
сервер +
↓
сохранено

```
if (invalidForms.length) {
  return { valid: false,
    formIds: invalidForms }
}
```

невалидная форма - **сохранение остановлено**

kaspersky

Три уровня проверки данных

1 Встроенные правила

Целый отдельный пакет с валидацией

```
"validation": {
  "rules": [
    { "method": "notEmpty" },
    { "method": "integer" }
  ]
}
```

notEmpty range integer …

33 правила в библиотеке

2 Кастомные по имени

функция живёт в коде плагина, схема зовёт её по строке

```
// form.json
{ "method": "function",
  "functionName":
  "handlers.2fa.codeValidation" }
```

↓ runtime.getSync ↓

```
// код плагина
runtime.join(
  'handlers.2fa.codeValidation',
  validate2FACode)
```

3 Серверная проверка

при сохранении клиентской валидации никогда не достаточно

форма
↓
клиент +
↓
сервер +
↓
сохранено

```
if (invalidForms.length) {
  return { valid: false,
    formIds: invalidForms }
}
```

невалидная форма - **сохранение остановлено**

kaspersky

Три уровня проверки данных

1 Встроенные правила

Целый отдельный пакет с валидацией

```
"validation": {
  "rules": [
    { "method": "notEmpty" },
    { "method": "integer" }
  ]
}
```

notEmpty range integer …

33 правила в библиотеке

2 Кастомные по имени

функция живёт в коде плагина, схема зовёт её по строке

```
// form.json
{ "method": "function",
  "functionName":
  "handlers.2fa.codeValidation" }
```

↓ runtime.getSync ↓

```
// код плагина
runtime.join(
  'handlers.2fa.codeValidation',
  validate2FACode)
```

3 Серверная проверка

при сохранении клиентской валидации никогда не достаточно

форма
↓
клиент +
↓
сервер +
↓
сохранено

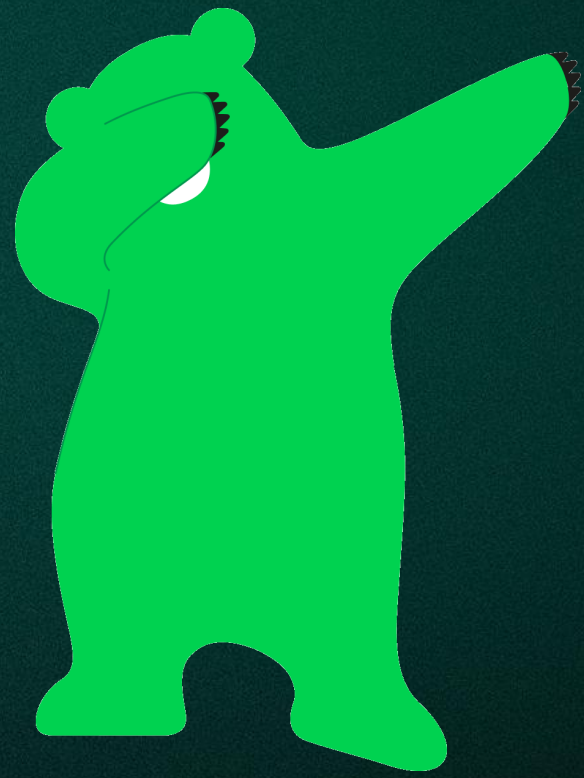
```
if (invalidForms.length) {
  return { valid: false,
    formIds: invalidForms }
}
```

невалидная форма - **сохранение остановлено**

kaspersky

Взросление формы

Глава третья



Дизайн-системы умирают.

Полторы тысячи форм должны продолжать работать.



Три поколения, один DSL

Riot

Историческое

Так сложилась жизнь. Компоненты живут и работают до сих пор.

Три поколения, один DSL

Riot

Историческое

Так сложилась жизнь. Компоненты живут и работают до сих пор.

Prev UI Kit

Второе поколение

Пришёл React - появился слой адаптеров к нему.

Три поколения, один DSL

Riot

Историческое

Так сложилась жизнь. Компоненты живут и работают до сих пор.

Prev UI Kit

Второе поколение

Пришёл React - появился слой адаптеров к нему.

New UI Kit

Текущее

Новая дизайн-система, на которую мы переезжаем сейчас.

Три поколения, один DSL

Riot

Историческое

Так сложилась жизнь. Компоненты живут и работают до сих пор.

Prev UI Kit

Второе поколение

Пришёл React - появился слой адаптеров к нему.

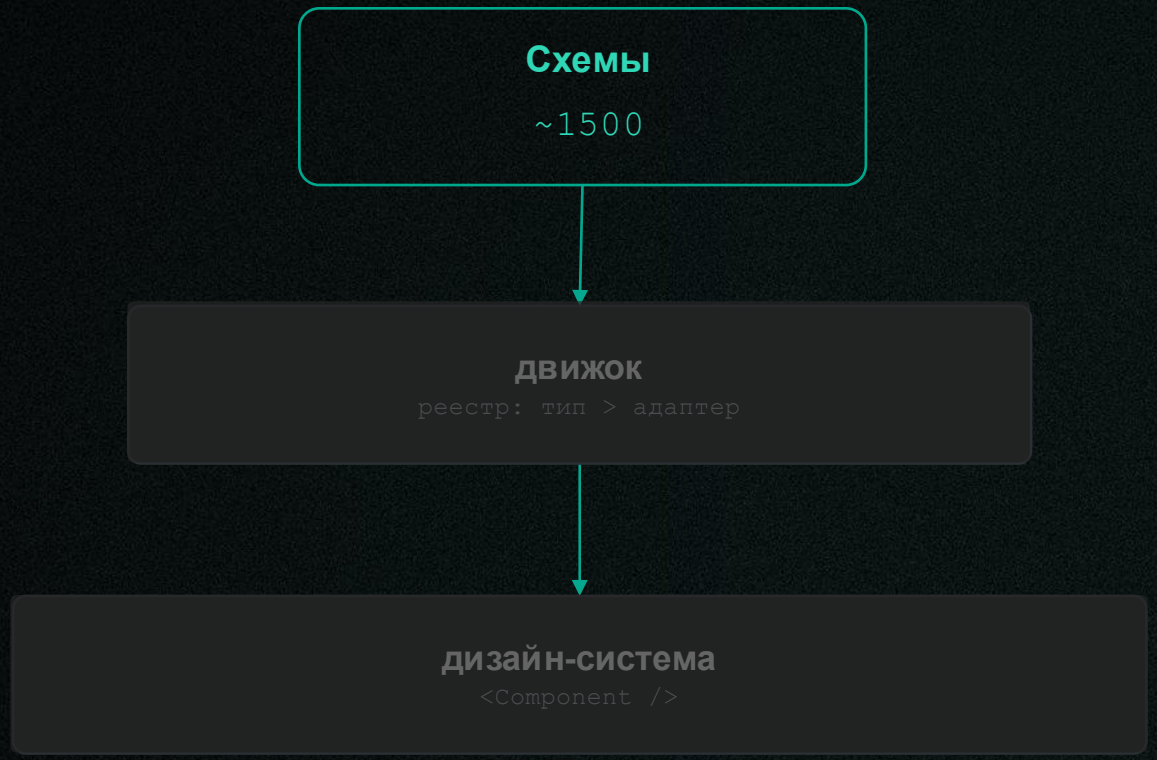
New UI Kit

Текущее

Новая дизайн-система, на которую мы переезжаем сейчас.

одна и та же схема

Переписать 1500 форм разом нельзя.
Остановить продуктовую разработку ради миграции - тем более.



activation-rule.json (схема формы)

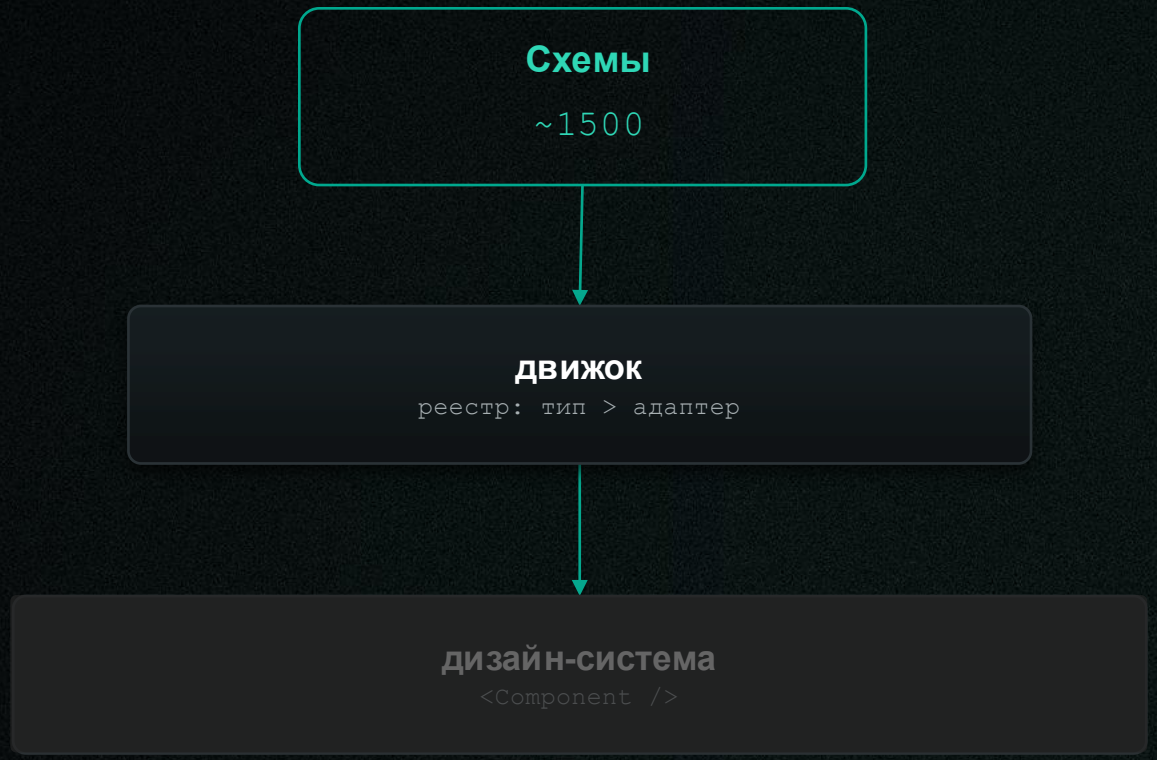
```
"type": "select",
"id": "operation",
"elements": [{ "value": 7, "text": "=" },
             { "value": 8, "text": "≠" }]
```

adapters/react/v6/index.ts (реестр типов)

```
export * from './select'
export * from './checkbox'
export * from './table'
// …около сорока типов
```

adapters/react/v6/select.tsx (адаптер)

```
import { Select as SelectComponent } from '@kl/hexa-ui'
…
<SelectComponent … />
```



activation-rule.json (схема формы)

```
"type": "select",
"id": "operation",
"elements": [{ "value": 7, "text": "=" },
             { "value": 8, "text": "≠" }]
```

adapters/react/v6/index.ts (реестр типов)

```
export * from './select'
export * from './checkbox'
export * from './table'
// …около сорока типов
```

adapters/react/v6/select.tsx (адаптер)

```
import { Select as SelectComponent } from '@kl/hexa-ui'
…
<SelectComponent … />
```

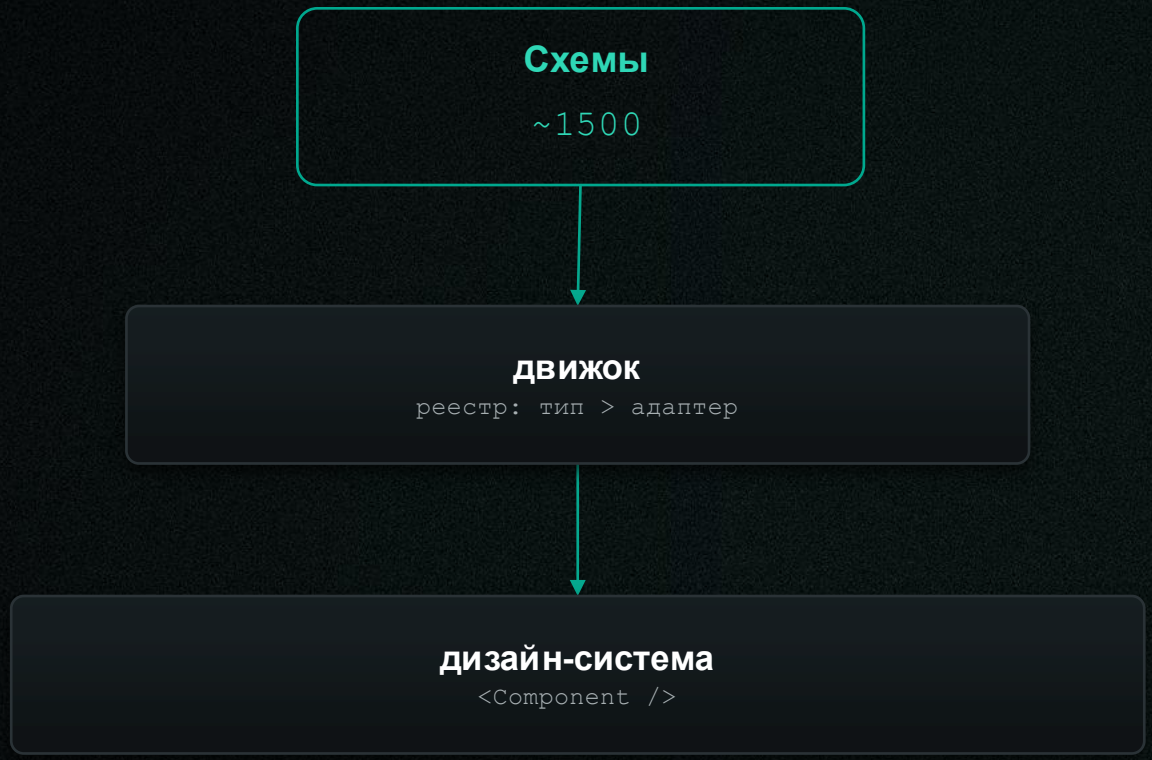


схема знает, **что** показать - движок знает, **как**

activation-rule.json (схема формы)

```
"type": "select",
"id": "operation",
"elements": [{ "value": 7, "text": "=" },
             { "value": 8, "text": "≠" }]
```

adapters/react/v6/index.ts (реестр типов)

```
export * from './select'
export * from './checkbox'
export * from './table'
// …около сорока типов
```

adapters/react/v6/select.tsx (адаптер)

```
import { Select as SelectComponent } from '@kl/hexa-ui'
…
<SelectComponent … />
```

Схема не знает, каким компонентом её рисуют. Именно поэтому UI kit можно менять.

Меняется слой, а не форма

domen.overview.json (поля в схеме)

```
{
  "type": "loader",
  "componentLibrary": "v1",
  "componentLibraryNext": "v2"
}
```

1 В схеме два поля: чем рисовать сейчас и чем - в будущем

Меняется слой, а не форма

v6-feature-flags.ts (выключатели)

```
const stableNewLibComponents = {
  button: true,
  checkbox: true,
  select: true,
  'control-group': false,   // ещё не переехал
}

const result = isNewComponentStable && isNewLibEnabled
```

1. В схеме два поля: чем рисовать сейчас и чем - в будущем
2. Движок смотрит на них и на фича-флаги

Меняется слой, а не форма

lib.ts

```
if (isNewLibFeatureFlagsEnabled) {
    lib = model.componentLibraryNext
  } else {
    lib = 'old-lib' // флаги выключены - откат
  }
```

1. В схеме два поля: чем рисовать сейчас и чем - в будущем
2. Движок смотрит на них и на фича-флаги
3. Адаптер раскладывает данные по параметрам компонента

Меняется слой, а не форма

lib.ts

```
if (isNewLibFeatureFlagsEnabled) {
    lib = model.componentLibraryNext
  } else {
    lib = 'old-lib' // флаги выключены - откат
  }
```

1. В схеме два поля: чем рисовать сейчас и чем - в будущем
2. Движок смотрит на них и на фича-флаги
3. Адаптер раскладывает данные по параметрам компонента

Форму при этом никто не трогает - меняется слой под ней.

Арсенал совместимости

1

Адаптеры и запасные варианты

нет нового компонента - рисуем старым

Арсенал совместимости

1

Адаптеры и запасные варианты

нет нового компонента - рисуем старым

2

Фича-флаги

новое поведение включается постепенно

Арсенал совместимости

1

Адаптеры и запасные варианты

нет нового компонента - рисуем старым

2

Фича-флаги

новое поведение включается постепенно

3

Белые списки

сложные компоненты - по списку форм

Арсенал совместимости

1

Адаптеры и запасные варианты

нет нового компонента - рисуем старым

2

Фича-флаги

новое поведение включается постепенно

3

Белые списки

сложные компоненты - по списку форм

4

Миграционные скрипты

таска переписывает схемы массово

Арсенал совместимости

1

Адаптеры и запасные варианты

нет нового компонента - рисуем старым

2

Фича-флаги

новое поведение включается постепенно

3

Белые списки

сложные компоненты - по списку форм

4

Миграционные скрипты

таска переписывает схемы массово

5

Документация DSL

иначе через три года не помнит никто

Арсенал совместимости

1

Адаптеры и запасные варианты

нет нового компонента - рисуем старым

2

Фича-флаги

новое поведение включается постепенно

3

Белые списки

сложные компоненты - по списку форм

4

Миграционные скрипты

таска переписывает схемы массово

5

Документация DSL

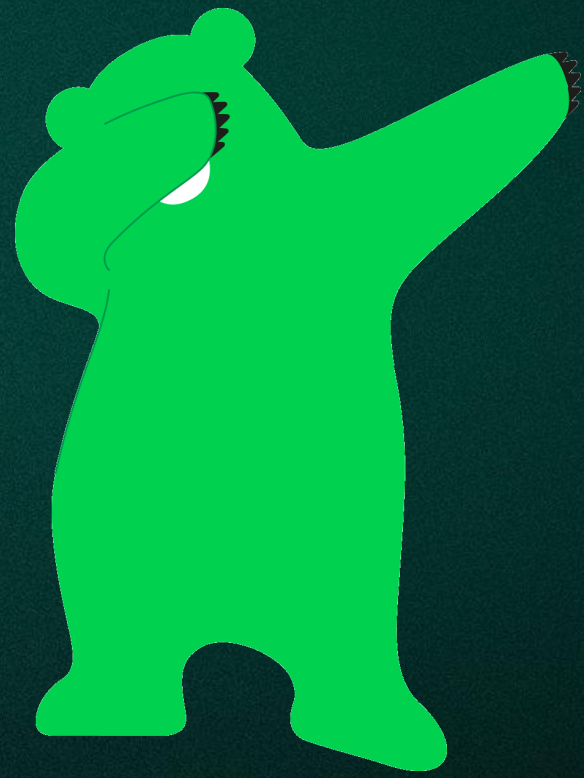
иначе через три года не помнит никто

Документацию тоже можно поручить генерации AI

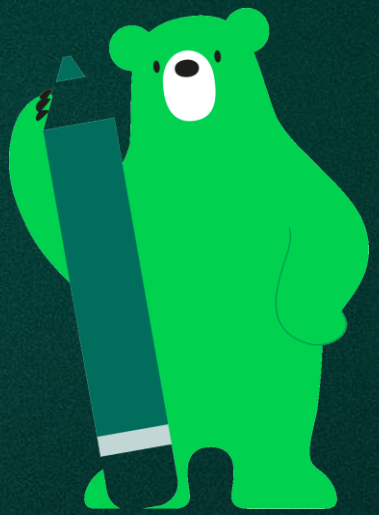


Долголетие формы

Глава четвертая



Много лет в проде.
Как DSL продолжать работать, а нам не свихнуться?



У DSL должен быть жизненный цикл, как у любого публичного API.



Версия

У DSL должен быть жизненный цикл, как у любого публичного API.



Версия

Changelog

У DSL должен быть жизненный цикл, как у любого публичного API.



Версия

Changelog

Политика удаления полей

У DSL должен быть жизненный цикл, как у любого публичного API.



- Версия
- Changelog
- Политика удаления полей
- Тесты совместимости

У DSL должен быть жизненный цикл, как у любого публичного API.

Версия

Changelog

Политика удаления полей

Тесты совместимости

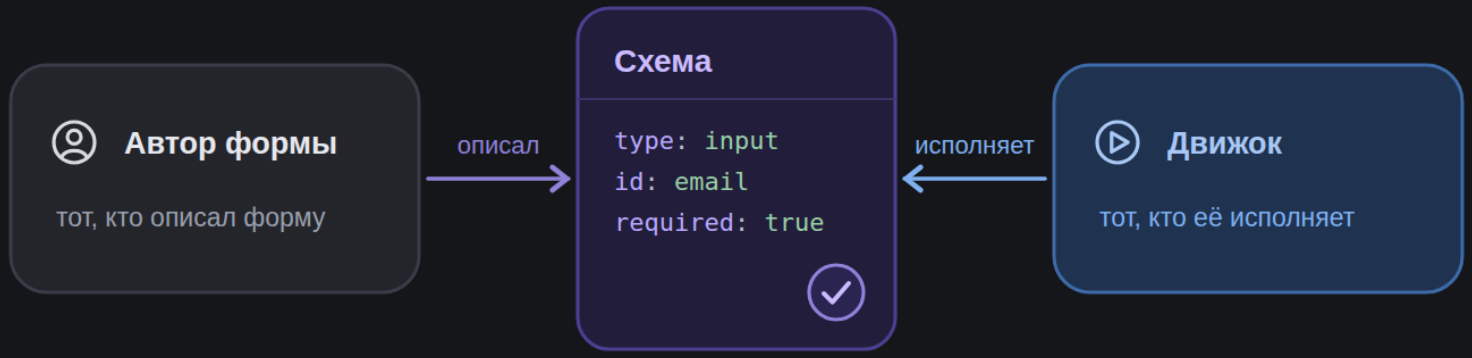


Схема — это **контракт** между тем, кто форму описал, и движком, который её исполняет.

Часть 5

Что меняет AI



Форма - это JSON-схема

form.json - фрагмент

```
{
  "type": "textbox",                          // тип элемента
  "id": "accountName",                        // имя поля
  "title": "Имя учётной записи",              // подпись
  "validation": {
    "rules": [                    // правила проверки
      { "method": "notEmpty" }
    ]
  }
}
```

>

Форма, собранная движком

Имя учётной записи

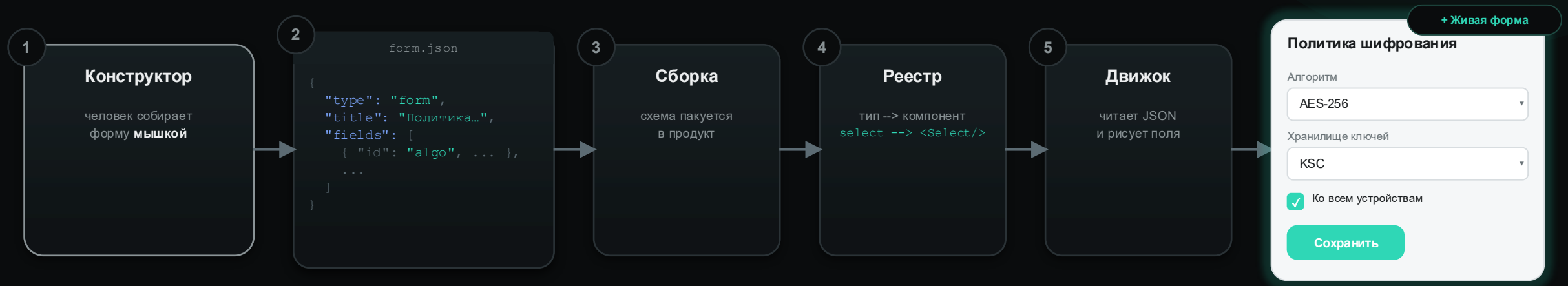
Введите имя…

Обязательное поле notEmpty

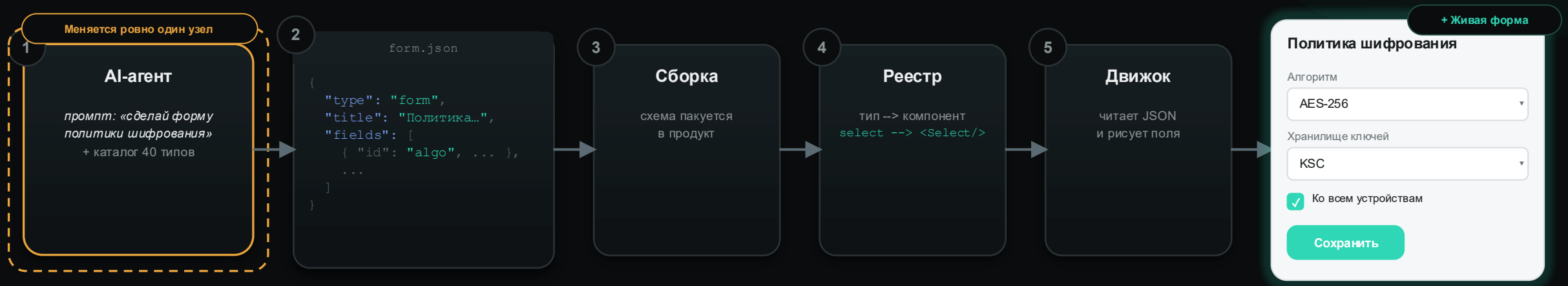
Описание

Основная учётная запись

Стандартный путь формы

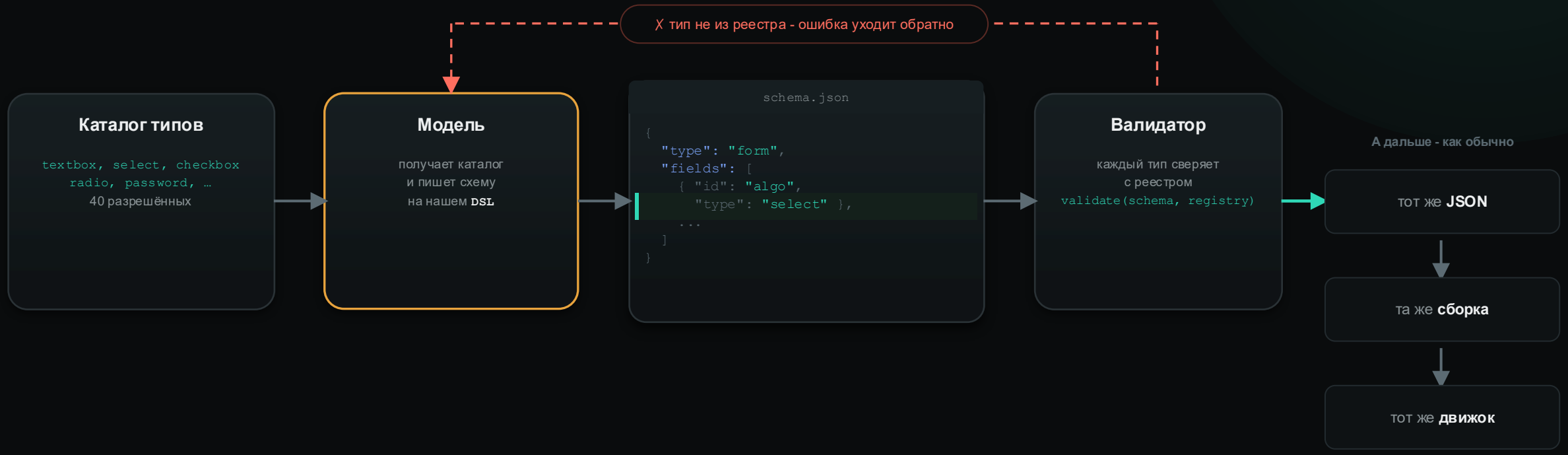


Тот же путь, но с AI



Прототип рабочий - всё прогнано руками. **Пара форм уже собрана агентом на dev, дальше - внедрение в плагины.**

Выдумала тип - чинит сама



проверка машинная: сверка со словарём - **один lookup, без магии**

kaspersky

Тот же путь, но автор - AI

1. **dsl-prompt.md** - инструкция для агента: каталог из всех разрешённых типов, правила, промпт
2. **agent.js**: вызывает модель (Cursor CLI), та пишет JSON-инструкцию. Кода - ноль
3. **check-form.js** - валидатор: все детали из каталога? id без дублей? зависимости ссылаются на существующее?

\+ 4. **forms/AI - ….json** – Готовая форма через ИИ

\+ 5. **Sandbox** – просмотр живой формы в изоляции

Язык, проверки и движок **уже были в продукте** - мы добавили одно: переводчика с человеческого

Мы не учим модель писать интерфейсы. Мы отдаём ей набор правил и ограничений.

Шаг 1. dsl-prompt.md — инструкция для агента

dsl-prompt.md

Скармливается модели **перед каждым запросом**. Как переводчику дают словарь: пользуйся только этими словами.

```
## Каталог типов ( только ограниченные типы из каталога )

header    Заголовок секции
textbox   Однострочный ввод
select    Выпадающий список
toggle    Переключатель
checkbox  Флажок
..., плюс анатомия элемента и полный пример
```

Выдумывать типы ЗАПРЕЩЕНО: схема проходит машинную проверку, неизвестный тип будет отклонён.

Шаг 2. agent.js – вызываем модель

agent.js

Обычный скрипт на 160 строк. Модель зовётся через **Cursor CLI** - работает на подписке Cursor, API-ключи не нужны.

```
for (let iter = 1; iter <= MAX_ITERATIONS; iter++) {
  const raw = await callModel(conversation)  // промпт --> модель
  schema = extractSchema(raw)              // вытащить JSON из ответа
  const errors = checkSchema(schema)       // отдать валидатору
  if (errors.length === 0) break           // если все ок – сохраняем форму
  conversation += «...ошибки... Исправь»       // если не ок - модель чинит сама
}
```

Кода интерфейса здесь ноль. Только оркестрация: склеить, спросить, проверить, вернуть ошибку.

Агент в терминале

node agent.js

$ node agent.js

▸ Промпт: «Сделай форму политики шифрования…»

Движок: Cursor CLI, в контексте: документация DSL + каталог типов

▸ Итерация 1: модель генерирует схему…

ответ за 51 с, 3322 символов

✓ Схема прошла машинную проверку (9 элементов)

✓ Схема сохранена: forms/AI - Encryption policy.json

▸ Test Flight: http://localhost:3001/#!/forms/preview/encryption-policy

Шаг 3. Проверяющий

check-form.js

Тупая программа на 130 строк: **не ошибается и не «соглашается»**. Список типов скопирован из исходников движка.

```
const REGISTRY = ['accordion', 'button', 'checkbox',
  'select', 'textbox', 'toggle', /* …все типы */]

if (!REGISTRY.includes(el.type)) {
  errors.push(`тип "${el.type}" не из реестра.
    Разрешены: ${REGISTRY.join(', ')}`)  // этот текст уйдёт модели
}
```

Автофикс Это не свойство модели, а свойство контура: есть кому проверить и есть куда вернуть ошибку.

Валидатор в бою

Модель придумала тип «password-strength-meter» - красивый, логичный и несуществующий.

node agent.js --fix ai-hallucinated.json

```
$ node agent.js --fix ai-hallucinated.json

✗ Валидатор отклонил схему:
  элемент[6] "el7": тип "password-strength-meter" не из реестра.
  Разрешены: accordion, button, checkbox, …, textbox, toggle, …

▸ Итерация 2: модель чинит сама… ответ за 21 с
✓ Схема прошла машинную проверку - выдуманный тип заменён на легальный
```

× Свободная генерация

Всплывёт в рантайме, пустым местом на форме.

+ Спецификация

Ошибка уходит модели - она чинит сама.

Что понадобилось для автоматизации

Уже было в продукте

язык схем (DSL) и каталог из 40 типов

движок: схема --> живая форма

~1500 настоящих схем в репозитории

Sandbox для просмотра форм

Добавили

+ инструкция для модели - dsl-prompt.md
+ Чекер по каталогу- 130 строк
+ Оркестратор с самопочинкой - 160 строк
+ Ноль строк кода для интерфейса

AI не заменил платформу - **он заговорил на её языке**

Обвязка для агента оказалась той же обвязкой, которую мы много лет строили для людей.

Часть 6

Где ложка дёгтя?
Посмотрим честно



Как проект начинался

Опишем формы схемой - и заживём



Как проект начинался

Опишем формы схемой - и заживём



Как проект идёт

«Что значит вот эта штука в схеме?»



Что накапливается за годы

- Разные версии схем - писались в разные годы



Что накапливается за годы

- Разные версии схем - писались в разные годы
- Старые поля, которые нельзя удалить



Что накапливается за годы

- Разные версии схем - писались в разные годы
- Старые поля, которые нельзя удалить
- Кастомные настройки у каждого микрофронтенда



Что накапливается за годы

- Разные версии схем - писались в разные годы
- Старые поля, которые нельзя удалить
- Кастомные настройки у каждого микрофронтенда
- Миграции значений - смысл поля поменялся



Что накапливается за годы

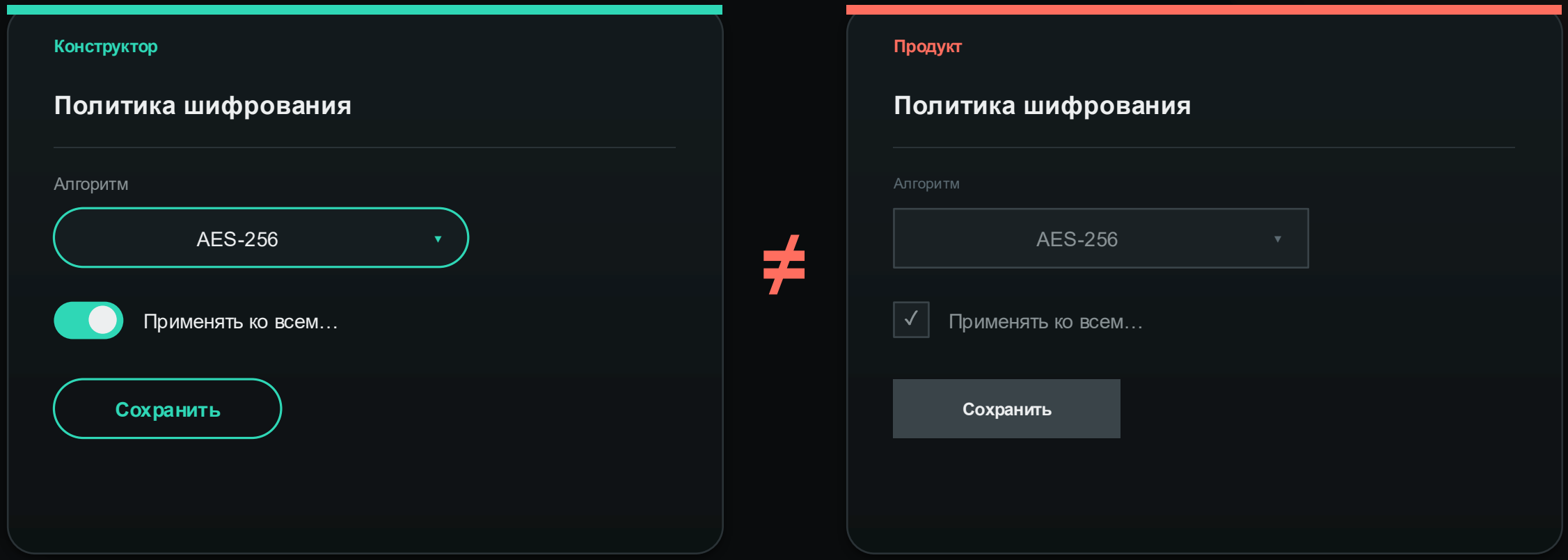
- Разные версии схем - писались в разные годы
- Старые поля, которые нельзя удалить
- Кастомные настройки у каждого микрофронтенда
- Миграции значений - смысл поля поменялся
- Неочевидные дефолты - магические числа

style - магические числа

```
"style": {
  "indent": 0,
  "offsetBottom": 0,
  "offsetTop": 0,
  "labelPos": 1,
  "labelType": 0
}
```

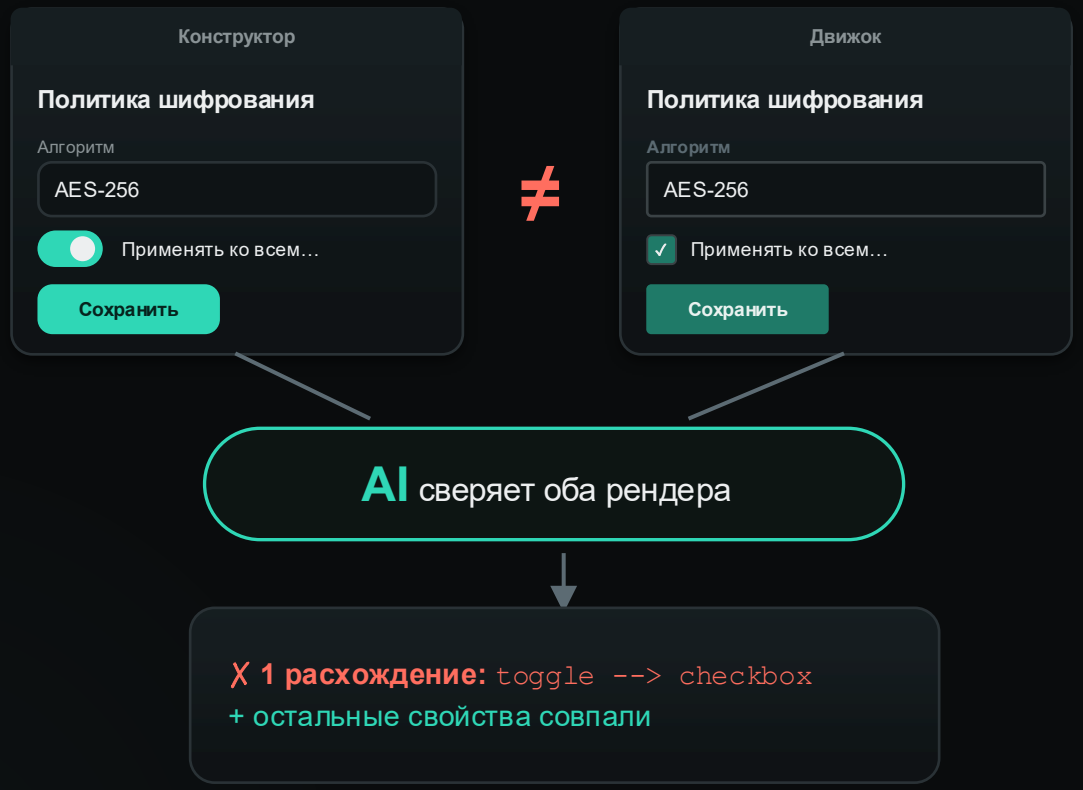
Что значит labelPos: 1? A labelType: 0?
Смысл знает только конструктор и движок.

Конструктор и движок разошлись.



Свойства новой дизайн-системы есть в конструкторе **но движок про них не знает**.

Проверку отдаём модели



Что ещё отдаём модели

Сверка расхождений

что конструктор и движок говорят об одном и том же - *как слева*

Магические числа

объяснит на лету: `keyLength: 128` --> минимальная длина ключа

Документация

черновик по коду и схемам - человек только вычитает

DSL становится огромным

AI не лечит

policy-form.json

```
{
  "type": "form",
  "id": "encryptionPolicy",
  "style": { "layout": "twoColumns", "gap": "m" },
  "fields": [
    {
      "id": "algo", "type": "select",
      "validation": { "rules": [{ "method": "notEmpty" }] },
      "visibleIf": { "field": "enabled", "eq": true },
      "dataSource": { "type": "staticFunction", "name": "getAlgos" },
      "featureFlag": "v6-select",
      ...
    }
  ]
}
```

Поверхность DSL

40	типов компонентов
~30	правил валидации
style	раскладка, отступы, колонки
visibleIf	зависимости между полями
flags	выключатели фич

Ограниченный словарь ≠ маленький язык - **поверхность DSL всё равно выросла**

Схемы с примесью бизнес-логики

AI не лечит

schema.json (фрагмент)

```
"dependencies": [
  { "field": "address",
    "visibleIf": { "useProxy": true } },
  { "field": "port",
    "visibleIf": { "useProxy": true },
    "requiredIf": { "address": "!empty" } },
  { "field": "auth",
    "visibleIf": { "useProxy": true,
                   "address": "!empty" } },
  ...
]
```

=

одно и
то же

чем это является на самом деле

```
if (useProxy) show(address)
if (useProxy) show(port)
if (useProxy && address) require(port)
if (useProxy && address) show(auth)
if (auth !== "none") show(login)
if (auth === "ldap") show(domain)
if (port < 1 || port > 65535) error(port)

... × 13 таких же, перекрёстных
```

Бизнес-логика в форме является результатом подхода

Одного React уже мало



Порог входа вырос - снижаем **документацией и примерами**

kaspersky

React есть на каждой работе. Наш DSL - только у нас.

Знает весь рынок

React | TypeScript | Vite | Playwright

+ общепринятый стек и знания

Знаем только мы

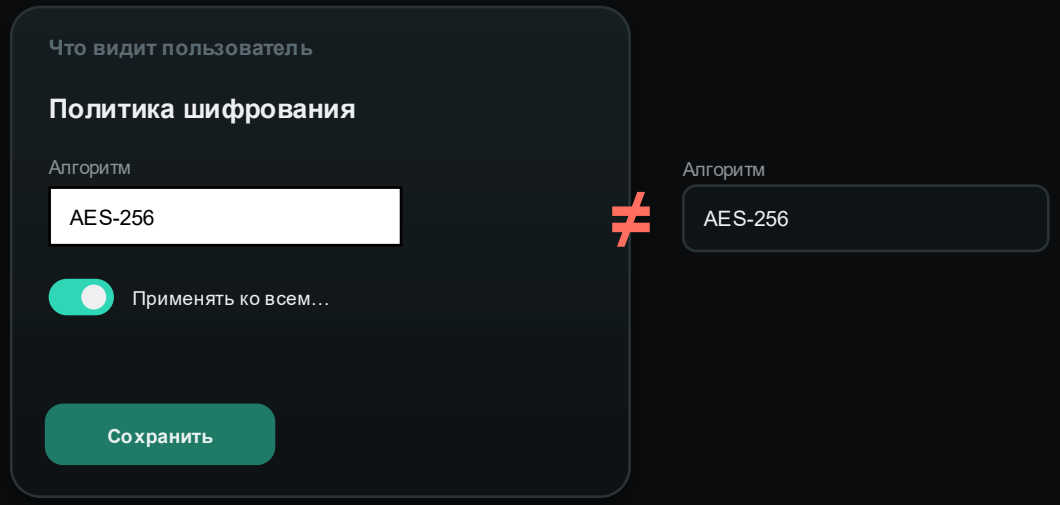
наш DSL | конструктор | buildUi + jsonpack | соглашения об именах

Сомнительный профит на внешнем рынке

> *Я изучаю технологию, которая не конвертируется на рынке*

Инструмент, которого нет на рынке, приходится **продавать внутри** - документацией, примерами и поддержкой.

Почему-то поехала вёрстка



Нет ~~stack trace компонента~~

привычный путь отладки оборван
компонент ни при чём - он даже не запускался

```
[engine] адаптер "select" не найден
[engine] переключился на старую библиотеку
[engine] форма отрисована +
```

½ дня **уходит на расследование**

отрисовка - минута,
остальное - поиск причины

Fail-open спасает пользователя - но прячет ошибку от разработчика

TypeScript про схему ничего не знает

schema.json

```
{
  "id": "encryptionPolicy",
  "fields": [
    { "id": "algo", "type": "select" },
    { "id": "keyLength",
      "type": "number",
      "min": 128 }
  ]
}
```

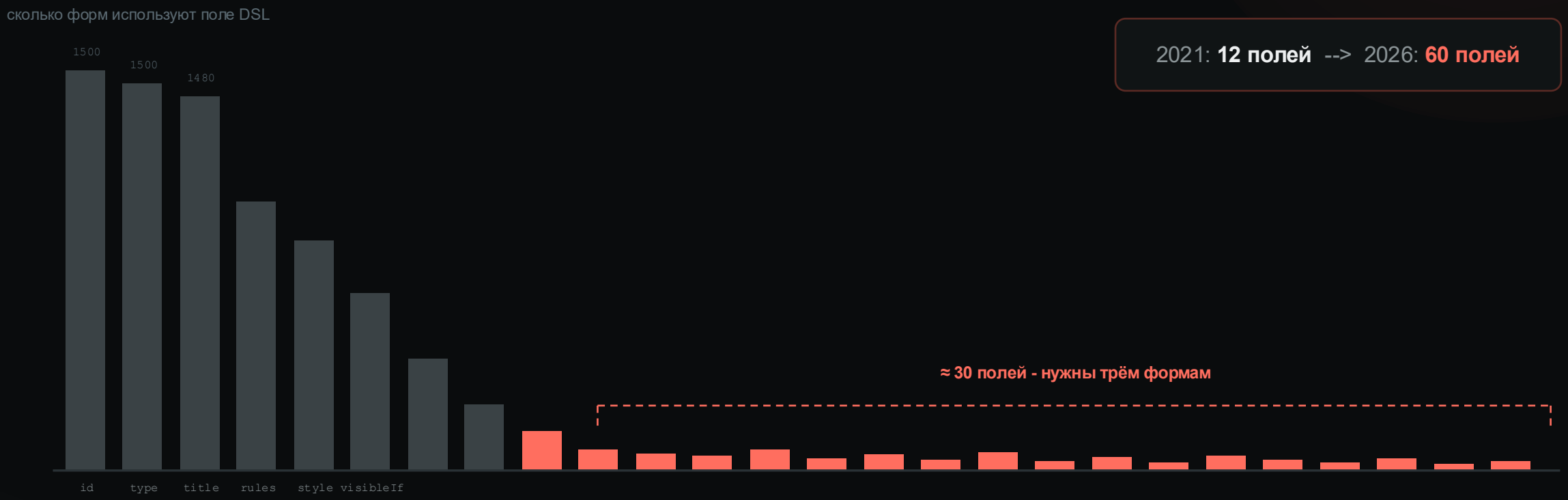
generate

form-types.ts

```
interface EncryptionPolicy {
  fields: {
    algo: { type: "select" };
    keyLength - типа нет
  };
}
```

Типы **генерируются из схем,** однако рассинхрон копится

Множество ненужных данных



2021: **12 полей** --> 2026: **60 полей**

Каждый кастомный кейс платит в общий словарь - **универсальность протекает по полю за раз**

Нечитаемые поля формы

```
{
  "buildType": "any",
  "componentLibrary": "riot",
  "componentLibraryNext": "v6",
  "isNotSortable": false,
  "selectedColumnsVecFieldsSyncEnvVarName": null,
  "selectionSyncEnvVarName": null,
}
```

```
  "buildType": { "slct": "any" },

  "when": { "type": "3", "value": false }
```

Множество форм содержат старый компонент



поменять поведение -
значит сломать 300 форм

Обратная совместимость - тормоз развития: отсюда **белые списки и постепенность**

Schema Driven UI - это продукт внутри продукта.

Документация

Schema Driven UI - это продукт внутри продукта.

Документация

Миграции

Schema Driven UI - это продукт внутри продукта.

Документация

Миграции

Тесты

Schema Driven UI - это продукт внутри продукта.

Документация

Миграции

Тесты

Команда

Когда нужно, а когда нет

+ Нужно

- Повторяемые сценарии: формы, настройки, CRUD
- Однотипных экранов 50 и больше, не 5
- Несколько команд, нужна консистентность
- Есть или планируется дизайн-система



Когда нужно, а когда нет

Нужно

- Повторяемые сценарии: формы, настройки, CRUD
- Однотипных экранов 50 и больше, не 5
- Несколько команд, нужна консистентность
- Есть или планируется дизайн-система

Не нужно

- Экранов мало и каждый уникален
- Нет ресурса на поддержку движка
- Нужен полный контроль над каждым пикселем



Когда нужно, а когда нет

Видим скорость, если DSL хорошо ограничен.
Начинает вредить, если в него складывают бизнес-логику.



С чего начать: шаги 1–3

1	**Найти сценарии**	Не все экраны, а конвейерные: настройки, CRUD-формы, мастера, фильтры. Три уникальных экрана - вам это не нужно, честно.

С чего начать: шаги 1–3

1	**Найти сценарии**	Не все экраны, а конвейерные: настройки, CRUD-формы, мастера, фильтры. Три уникальных экрана - вам это не нужно, честно.
2	**Минимальный набор**	Компоненты, покрывающие ~80% сценариев: поле, число, переключатель, список, дата, таблица, кнопка.

С чего начать: шаги 1–3

1	**Найти сценарии**	Не все экраны, а конвейерные: настройки, CRUD-формы, мастера, фильтры. Три уникальных экрана - вам это не нужно, честно.
2	**Минимальный набор**	Компоненты, покрывающие ~80% сценариев: поле, число, переключатель, список, дата, таблица, кнопка.
3	**Движок и реестр**	Это сердце, его пишут лучшие люди команды: дальше его будут использовать все, а менять - больно.

Шаги 4–5 и лазейка на всякий случай

4

Перевести и замерить

Возьмите самые типовые формы и замерьте эффект: время разработки, количество кода, число багов.



Шаги 4–5 и лазейка на всякий случай

4 Перевести и замерить

Возьмите самые типовые формы и замерьте эффект: время разработки, количество кода, число багов.

5 Оставить лазейку

С первого дня. У нас это react-wrapper: в схему можно вставить произвольный React-компонент, написанный руками.



Собираем - вместе с AI. Проектирует - человек.

Собираем с AI

движок AI + человек

рендерит схему в живую форму

первые формы AI + человек

черновики по каталогу из 40 типов

граница

Проектирует человек

реестр компонентов человек

что вообще существует

контракты человек

кто с кем и как говорит

границы DSL человек

что схеме можно - и что нельзя

Итог

Эту часть нельзя отдать на аутсорс - как бы хорошо ни писала модель.



Как внедрять, не останавливая продукт

Новые формы	только схемами, без исключений - иначе старый мир растёт быстрее нового

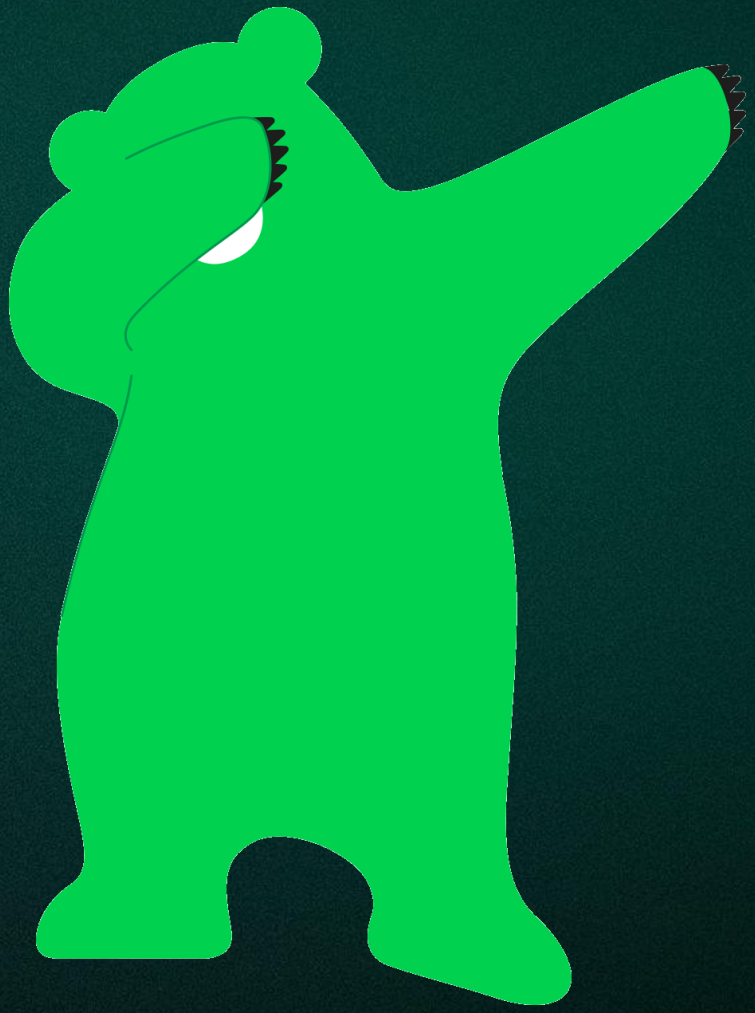
Как внедрять, не останавливая продукт

Новые формы	только схемами, без исключений - иначе старый мир растёт быстрее нового
Старые формы	переписываем по касательной: пришёл делать задачу - заодно переведи

Как внедрять, не останавливая продукт

Новые формы	только схемами, без исключений - иначе старый мир растёт быстрее нового
Старые формы	переписываем по касательной: пришёл делать задачу - заодно переведи
Массовые переходы	только скриптами, с автоматической проверкой результата

Schema Driven UI - это способ создать ограниченный язык для UI-платформы.



+ Три поколения UI,

+ 1500 форм,

+ переезд между микрофронтендами –

система продолжает работать.





AI делает архитектурную дисциплину обязанностью

Движок пишем вместе с AI, контракт проектируем сами

Реестр, контракты и границы DSL остаются за человеком

Архитектура

Enterprise-проблемы

Frontend

Платформы

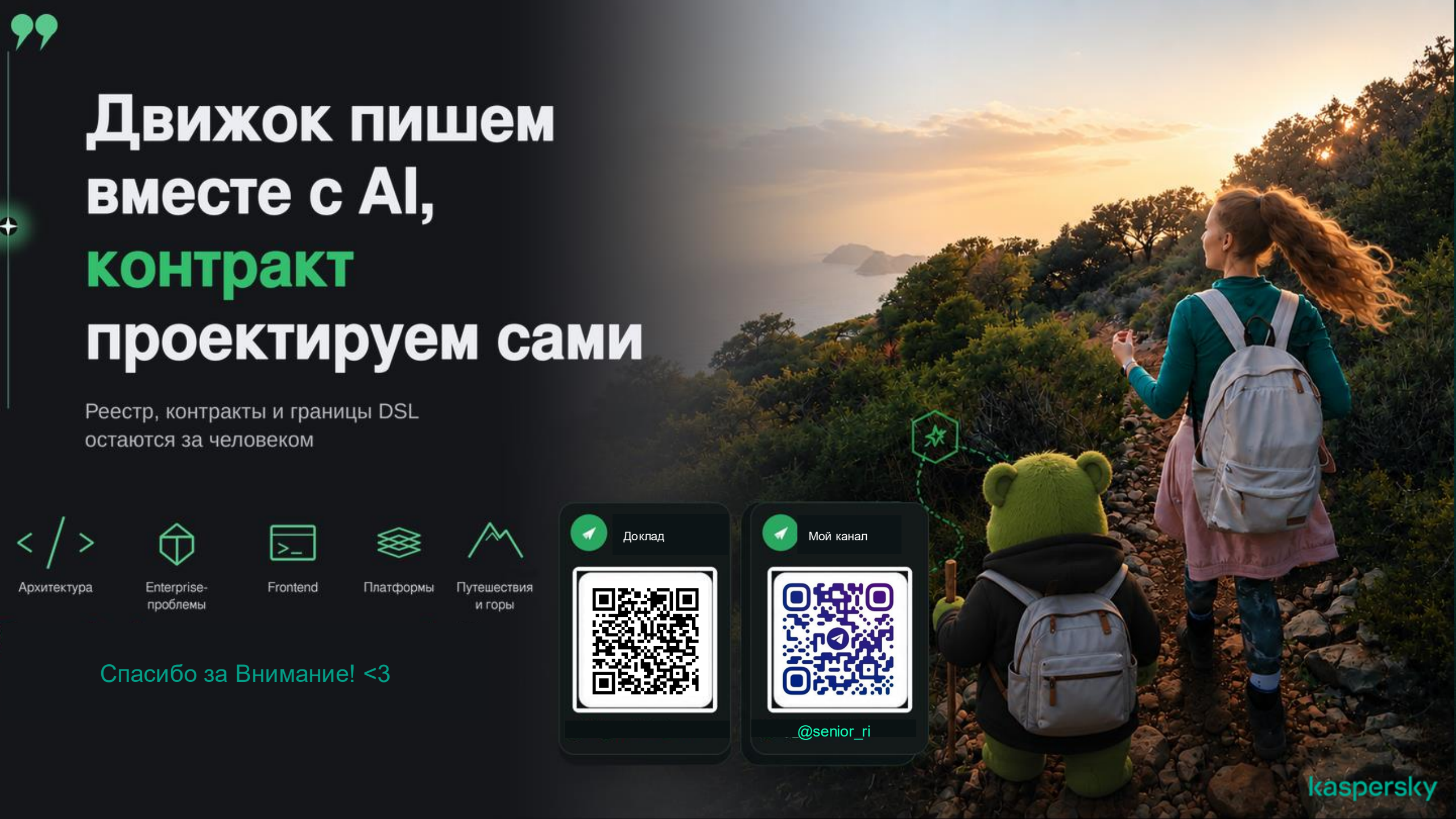
Путешествия и горы

Спасибо за Внимание! <3

Доклад

Мой канал

@senior_ri



kaspersky